НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 37.041

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: « *Інтерактивна система підтримки вивчення мови програмування JavaScript* »

Виконав: студент *4* курсу, групи *ІС-з51*
Даниленко Іван Іванович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доц., к.ф-м.н., доц. Гавриленко О.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Т.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *ст. викладач Шимкович В.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Даниленко І.І. _____
(підпис)

Київ – 2019 р.

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 28 рисунків, 12 таблиць, 1 додаток, 12 джерел.

Дипломний проект присвячений розробці інтерактивної системи підтримки вивчення мови програмування JavaScript. Цілі створення інтерактивної системи: допомога в вивченні мови програмування JavaScript, допомога в підготовці до співбесіди на front-end розробника. Для досягнення цілей у роботі вирішуються наступні задачі: створення вправ різного направлення для якісного вивчення мови JavaScript, перевірка відповіді до завдання на правильність, розподілення завдань по рівням складності.

У розділі загальних положень було наведено наведено словесний опис предметного середовища. Визначено призначення, цілі та задачі розробки застосування.

У розділі інформаційного забезпечення описано інформаційне забезпечення застосування, а саме, детально розглянута структура вхідних даних, які використовуються для подальшої обробки.

У розділі з програмного та технічного забезпечення було описано забезпечення, яке було використане при розробці застосування. Наведені діаграми функціонування програмного продукту та його структури.

У технологічному розділі було описано керівництво користувача. Пройдено випробування відповідності функцій застосування вимогам технічного завдання.

JAVASCRIPT, ВЕБ, РОЗРОБКА, FRONT-END, ES6, ВПРАВИ, ПРАКТИКА

ABSTRACT

The structure and scope of work. Explanatory note diploma project consists - of five sections containing 28 figures, 12 tables, 1 application, 12 sources.

Diploma project dedicated to the development of interactive JavaScript learning support system. Goals of the development of interactive system: assistance in learning the JavaScript programming language. help in preparing for an interview on the front-end developer. To achieve these goals are solved the following tasks: creating exercises for a qualitative study of the JavaScript language, checking the answer to the task for correctness, distributing tasks by the levels of complexity.

Under the general provisions were given a verbal description of the objective environment. Defined purpose, aims and objectives of system development. Also described basic processes of business and functional model.

In the information provision described information support systems, namely, reviewed in detail the structure of input data used for further processing.

The section of the software and hardware described software that has been used in the design of the system. These diagrams of the system and its structure.

In technological section described user manual. Passed the test match system functions requirements specification.

JAVASCRIPT, WEB, DEVELOPMENT, FRONT-END, ES6,
EXERCISES, PRACTICS

ЗМІСТ

ВСТУП

З самого початку існування інтернету, JavaScript був фундаментальною технологією, що створювала взаємодію користувача з сайтом, на якому він знаходився. Хоча спочатку JavaScript задумувався як спосіб відстежувати дії мишки та показувати вспливаючі повідомлення, більш як 2 десятиліття потому технологія і можливості JavaScript виросли на багато порядків, и зараз ніхто не сумнівається в його важливості для найбільш поширеної світової технології – інтернету.

Оскільки почати програмувати на JavaScript надзвичайно просто, достатньо лише блокнота та браузера, ця мова програмування стала надзвичайно доступною для широкої аудиторії розробників, навіть тих, що практично не мають ніякого досвіду в програмуванні. Написати «Hello world!» на JavaScript настільки просто, що мова приваблює та легко стає комфортною буквально з самого початку її вивчення.

Але, не дивлячись на те, що JavaScript можливо є одною з найлегших мов програмування для початківців, людей, що опанували цю технологію повністю зустрічається набагато менше ніж в інших мовах програмування. В порівнянні з такими мовами програмування як C чи C++, де для написання повномасштабної програми потрібно мати доволі глибокі знання мови, повномасштабний проект на JavaScript може, і часто й робить, лише торкається поверхні того, на що ця мова здатна.

Мій проект є актуальним, оскільки, в результаті аналізу доступних ресурсів для вивчення цієї мови програмування, я дійшов до висновку, що більшість із них приділяють JavaScript недостатньо уваги і не дозволяють в повній мірі пізнати цю технологію.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Об’єкт дослідження: «Інтерактивна система підтримки вивчення мови програмування JavaScript». Людей, які бажають вивчити мову JavaScript стає все більше. За результатами голосування від stackoverflow станом на 2019 рік JavaScript є найбільш популярною технологією серед користувачів цієї багатимільйонної спільноти. Результати голосування зображені на рисунку 1.1.

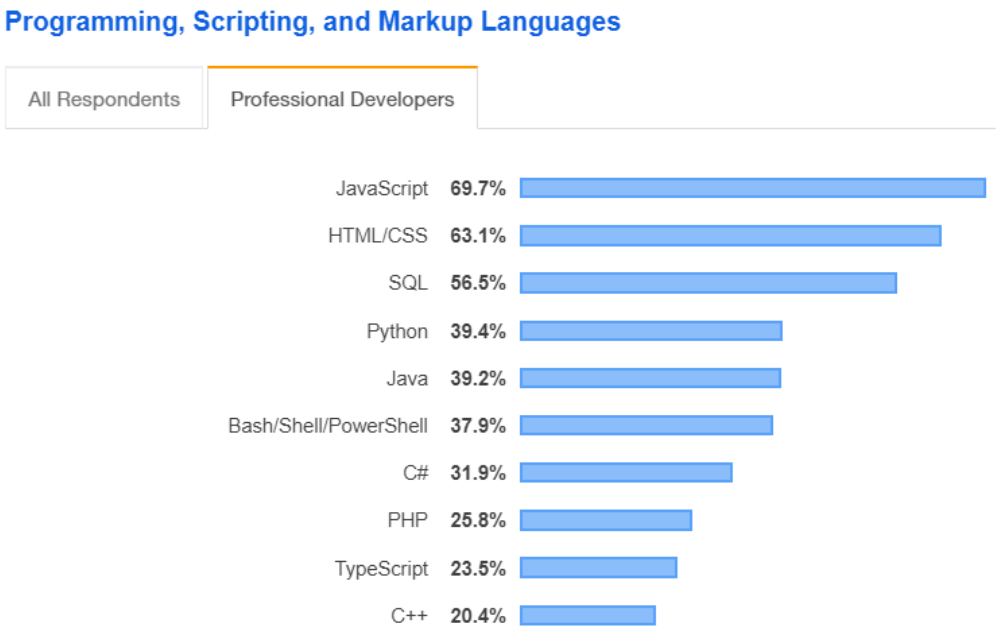

Рисунок 1.1. Популярні технології 2019

Постає питання в тому, яким чином організувати процесс вивчення, щоб він приносив максимальну користь. Більшість веб-ресурсів, що дозволяють вчити програмуванню, намагаються охопити всі популярні технології, в результаті чого дають інформацію лише на базовому рівні.

Поставлене мною завдання – це розробити таке програмне середовище, що дозволить створити глибоке та всестороннє розуміння мови програмування JavaScript, а також підготуватися до співбесід та отримати роботу в сфері веб-розробки.

1.1.1 Опис процесу діяльності

Процес розпочинається з прохання ввійти в систему або зареєструватися, якщо користувача ще немає в системі.

Далі користувач обирає одну з вправ:

– якщо користувач раніше проходив якісь завдання з обраної вправи, то система автоматично перенаправляє його на останню невиконану вправу, в іншому разі на першу;

– користувач виконує завдання та набуває практичних та теоретичних вмінь, що і є основним завданням для створення системи.

Інтерактивна система підтримки вивчення мови програмування JavaScript - це навчальна система, принцип якої полягає в вивченні мови програмування JavaScript за допомогою спеціально розроблених вправ. Кожна вправа має свою мету та розрахована на розвиток певних вмінь та розуміння мови JavaScript в цілому.

1.1.2 Опис функціональної моделі

Інтерактивна система підтримки вивчення мови програмування JavaScript дозволить всім web-розробникам полегшити процесс освоєння мови програмування JavaScript, що є незамінною при роботі з web-технологіями.

Інтерактивна система підтримки вивчення мови програмування JavaScript представляє собою веб-застосування, що дозволяє користуватися системою з любої точки світу через інтернет.

При користуванні веб-застосуванням система просить користувача зареєструватися чи війти в систему, якщо він вже має аккаунт. Це дозволить системі слідкувати за прогресом користувача для комфортного навчання.

Система пропонує користувачу на вибір 3 вправи, кожна вправа має свої унікальні особливості та розрахована на розвиток користувача в певному аспекті мови програмування JavaScript.

1.2 Огляд наявних аналогів

Наявні аналоги, розглянуті в цьому розділі є в більшості сайти, які орієнтуються на широкий набір вивчаємих мов програмування з різними принципами та структурою. Найбільш відомий аналог – це freecodecamp.org, всесвітній сайт для вивчення JavaScript. Але даний сайт також має певні недоліки, і перш за все – це незручність в користуванні, потрібно виконати не менш як 5 дій на сайті, перш ніж ти можеш попасти в тренувальну вправу, що є дуже незручним на мій погляд.

Інший аналог – це codewars.com. Але цей сайт більш розрахований на доволі високий рівень в знанні мови програмування (Мов програмування для вибору є дуже багато). А також сам сайт дуже нагромаджений якорними об’єктами, що дуже відволікає і не дає повністю вникнути в суть завдань та відкинути все зайве.

Також доволі відомий аналог – codecademy.com. Сайт зручний в використанні та має приємний вигляд. Головний недолік в тому, що сайт в основному платний, а в безкоштовному аккаунті користувач має доволі обмежені можливості.

З наявних аналогів можна зробити висновок, що моя система повинна бути швидкою, не нагромадженою зайвими дизайнерськими елементами та зручною в користуванні. Це і є головний напрямок подальшого розвитку проекту.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням системи є допомога в вивченні мови програмування JavaScript.

1.3.2 Цілі та задачі розробки

Цілі створення системи:

– допомога при вивченні мови програмування JavaScript;
– підготовка до співбесіди на front-end розробника;
– можливість швидко повторити конструкції мови JavaScript.

Для досягнення поставлених цілей необхідно вирішити наступні задачі:

- розробити вправу для практичної роботи з кодом JavaScript при вирішенні задач різних рівнів складності;
- розробити вправу, що покращує навик читання коду і розуміння того, як цей код виконається;
- розробити вправу для покращення знання теоретичних основ мови.

Висновок до розділу

В даному розділі були описані мета та цілі розробки, створені базові вимоги, які потрібно виконати при розробці програми. Було описано значення даної програми для користувача, наведені найближчі аналоги та порівняльний аналіз їх з даним програмним продуктом.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані вводяться в систему адміністратором.

Адміністратор може вводити наступні дані:

– інформацію про користувача;
– опис, приклад коду, умови та тести для виконання завдання з вправи «Тренуй себе»;
– запитання, приклад коду, відповідь для виконання завдання з вправи «Визнач результат»;
– запитання, варіанти відповіді для завдання з вправи «Перевір знання».

2.2 Вихідні дані

Вихідними даними системи є результати виконання завдань, прогрес користувача, а також статистичні данні про роботу системи.

2.3 Опис структури бази даних

В основі зберігання даних була використана база даних LokiJS, що розроблена на мові JavaScript та може зберігати в собі колекції даних в вигляді масивів інформації.

Масиви інформації в вигляді XML розмітки представлені на рисунках 2.1-2.4.

```xml
<users>
    <user>
        <id>1</id>
        <login>test</login>
        <password>test</password>
        <token>test-jwt-token</token>
        <progress>
            <ex1Level>0</ex1Level>
            <ex1Score>0</ex1Score>
            <ex2Level>0</ex2Level>
            <ex2Score>0</ex2Score>
            <ex3Level>0</ex3Level>
            <ex3Score>0</ex3Score>
        </progress>
    </user>
    <user>
        <id>2</id>
        <login>demo</login>
        <password>demo</password>
        <token>demo-jwt-token</token>
        <progress>
            <ex1Level>0</ex1Level>
            <ex1Score>0</ex1Score>
            <ex2Level>0</ex2Level>
            <ex2Score>0</ex2Score>
            <ex3Level>0</ex3Level>
            <ex3Score>0</ex3Score>
        </progress>
    </user>
</users>
```

Рисунок 2.1. Колекція користувачів

```
<ex1>
    <levels>
        <level>
        <id>1</id>
        <name>Рівень 1</name>
        <tasks>
            <task>
                <id>1</id>
                <name>Завдання 1</name>
                <description>{HTML розмітка}</description>
                <codeEditor>
                    <id>1</id>
                    <text>{Початковий JavaScript код}</text>
                </codeEditor>
                <challenges>
                    <challenge>
                    <id>1</id>
                    <title>{Умова для проходження тестів}</title>
                    </challenge>
                    <challenge>
                    <id>2</id>
                    <title>{Умова для проходження тестів}</title>
                    </challenge>
                </challenges>
                <tests>
                    <test>
                    <id>1</id>
                    <value>{Тест для перевірки}</value>
                    </test>
                    <test>
                    <id>2</id>
                    <value>{Тест для перевірки}</value>
                    </test>
                </tests>
                </task>
            </tasks>
        </level>
    </levels>
</ex1>
```

Рисунок 2.2. Масив завдань для вправи «Тренуй себе»

```
<ex2>
    <levels>
        <level>
            <id>1</id>
            <name>Рівень 1</name>
            <tasks>
                <task>
                    <id>1</id>
                    <name>Завдання 1</name>
                    <question>{HTML розмітка}</question>
                    <codeEditor>
                    <id>1</id>
                    <text>{JavaScript код}</text>
                    </codeEditor>
                    <answer>{Відповідь до завдання}</answer>
                </task>
            </tasks>
        </level>
    </levels>
</ex2>
```

Рисунок 2.3. Масив завдань для вправи «Визнач результат»

```
<ex3>
    <levels>
        <level>
            <id>1</id>
            <name>Рівень 1</name>
            <tasks>
                <task>
                <id>1</id>
                <name>Завдання 1</name>
                <question>{HTML розмітка}</question>
                <choises>
                    <choise>
                    <id>1</id>
                    <name>{Варіант відповіді}</name>
                    </choise>
                    <choise>
                    <id>2</id>
                    <name>{Варіант відповіді}</name>
                    </choise>
                </choises>
                <answer>{Відповідь до завдання}</answer>
                </task>
            </tasks>
        </level>
    </levels>
</ex3>
```

Рисунок 2.4. Масив завдань для вправи «Перевір знання»

Висновок до розділу

У даному розділі було розглянуто вхідні та вихідні дані для інтерактивної системи підтримки вивчення мови програмування JavaScript. Структура масивів інформації відповідає тим задачам, які повинна вирішувати система.

3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Програмне забезпечення, що використовується при розробці:

– платформа: Windows, Linux;
– веб-браузер: Google Chrome, Firefox;
– інструмент для дизайну: Figma;
– веб-сервер: Node.js, версія 10.15.1;
– менеджер пакетів: npm, версія 6.4.1;
– база даних: LokiJS, версія 1.5.6;
– система контролю версій: Git, версія 2.20.1;
– сервіс збереження git репозиторія: GitHub;
– середовище розробки: Visual Studio Code;
– препроцесор стилів: SASS, синтаксис SCSS;
– фреймворк для розробки клієнтської сторони: Angular 7;
– мови написання коду програми: JavaScript, TypeScript;
– використані бібліотеки: Ace, Express.

JavaScript – динамічна, об'єктно орієнтована, прототипна мова програмування загального призначення. Реалізація стандарту ECMAScript. Може використовуватися для: написання сценаріїв веб-сторінок для надання їм інтерактивності, створення односторінкових веб-застосунків (React, Angular, Vue.js), програмування на стороні сервера (Node.js), стаціонарних застосунків (Electron), мобільних застосунків (React Native, Cordova) та ін.

TypeScript – сумісна з JavaScript мова програмування, особливістю якої є чітка типізація. TypeScript позиціонується як засіб розробки веб-застосунків з покращенням швидкості розробки, розумінням, рефакторингом та повторним використанням коду.

Figma – інструмент для створення дизайну. Альтернатива для Adobe XD, Sketch та ін. Працює в браузері як веб-додаток та на комп’ютері

користувача. Зручний засіб для створення інтерактивних прототипів та користувацького інтерфейсу.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Надає можливість виконувати JavaScript-скрипти на сервері та відправляти результат їх виконання.

LokiJS – документно орієнтована база даних, написана на JavaScript, мета якої збереження JavaScript об’єктів як документи в не sql форматі. Працює в середовищах node.js, nativescript та браузері. Особливості LokiJs: швидка продуктивність, динамічні перегляди для швидкого доступу до підмножин даних, гнучкість в налаштуваннях та можливість додавання власної функціональності.

Git – одна з найефективніших, надійних, і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, які базуються на відгалудженні та злитті гілок. Надає можливість відновити будь-який зафіксований стан системи.

GitHub – один з найвідоміших та найпопулярніших сервісів для збереження git репозиторіїв. Також має можливість віддаленого злиття гілок, перегляду змін в коді системи та багато іншого.

Visual Studio Code – популярний та багатофункціональний редактор коду, написаний на JavaScript, що розповсюджується безкоштовно та доступний для платформ Windows, Linux і Mac OS.

SASS – скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS). Призначена для підвищення рівня абстракції коду та спрощення файлів CSS. Мова SASS має два синтаксиси: SASS - характеризується відсутністю фігурних дужок та крапок з комою та SCSS – використовує фігурні дужки подібно до CSS.

Angular 7 – написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій.

Ace – це редактор коду, написаний мовою JavaScript. Створений з метою отримати редактор коду, який працював би у веб-браузері, а також поєднував у собі зручність і швидкість нативних редакторів коду, таких як TextMate, Vim або Eclipse.

Express – програмний каркас розробки веб-застосунків для Node.js, спроектований для створення веб-застосунків і API. Має великий набір функціональності, мінімалістичний та дуже швидкий.

3.2 Вимоги до технічного забезпечення

3.2.1 Загальні вимоги

Вимогами до технічного забезпечення є:

- платформа: Windows, Linux;
- веб-сервер NodeJS, версія 10.15.1 або вище;
- менеджер пакетів: npm, версія 6.4.1 або вище;
- веб-браузер: Google Chrome, Yandex Browser чи Firefox (останні 2 версії).

3.3 Архітектура програмного забезпечення

3.3.1 Опис структури програмного забезпечення

В ході розробки «Інтерактивна системи підтримки вивчення мови програмування JavaScript» була поділена на два незалежні розділи: front-end частину та back-end частину.

Front-end частина побудована на фреймфорку Angular і відповідає за отримання та опрацювання даних, що надходять з сервера.

Back-end частина побудована в форматі REST сервера за допомогою технологій Node.js та фреймфорку Express і відповідає за роботу з базою даних LokiJS та відправлення запрошених даних.

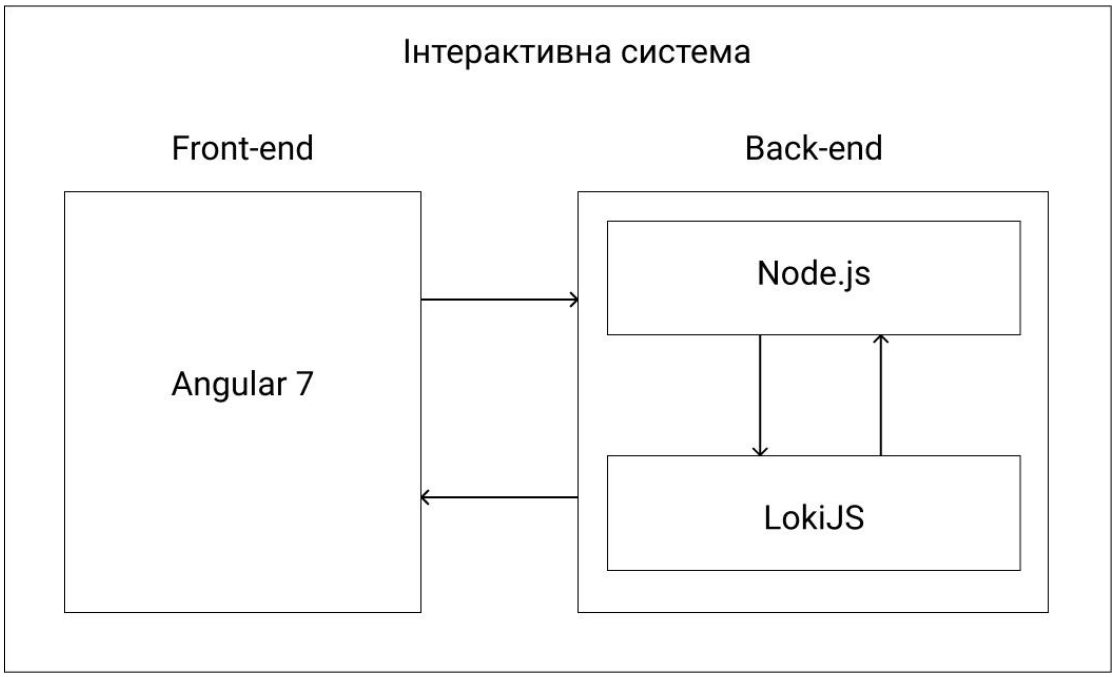

Рисунок 3.1. Структура програмного забезпечення

3.3.2 Front-end частина

3.3.2.1 Компоненти Angular

В ході роботи були створені наступні Angular компоненти:

– AppComponent – головний Angular компонент;
– HomeComponent – компонент, що відповідає за відображення головної сторінки;
– HeaderComponent – компонент, що відповідає за відображення шапки сайту;
– FooterComponent – компонент, що відповідає за відображення підвалу сайту;
– LoginComponent – компонент, що відповідає за вхід до системи;
– RegisterComponent – компонент, що відповідає за реєстрацію в системі;

- CabinetComponent – компонент, що відповідає за персональний кабінет користувача;
- ExercisesAllComponent – компонент, що відповідає за всі вправи;
- ExercisesListComponent – компонент, що відповідає за відображення списку вправ;
- SidebarComponent – компонент, що відповідає за
- Ex1Component – компонент, що відповідає за вправу «Тренуй себе».
- Task1Component – компонент, що відповідає за відображення пояснень та умов для виконання завдання з вправи «Тренуй себе».
- Answer1Component – компонент, що відповідає за редактор коду для виконання вправи «Тренуй себе».
- Ex2Component – компонент, що відповідає за вправу «Визнач результат».
- Task2Component – компонент, що відповідає за відображення опису завдання для виконання вправи «Визнач результат».
- Answer2Component – компонент, що відповідає за текстове поле для виконання вправи «Визнач результат».
- Ex3Component – компонент, що відповідає за вправу «Перевір знання».
- Task3Component – компонент, що відповідає за відображення опису завдання для виконання вправи «Перевір знання».
- Answer3Component – компонент, що відповідає за відображення списку з вибором відповідей для виконання вправи «Перевір знання».
- AdditionallyComponent – компонент, що відповідає за відображення ресурсів, на яких також можна вчити програмування на мові JavaScript.

Загальна діаграма компонентів зображена на рисунку 3.2.

Рисунок 3.2. Загальна діаграма компонентів

Діаграми компонентів залежно від маршруту в url адресі представлені на рисунках 3.3-3.11.

url адреса – http://localhost:4200

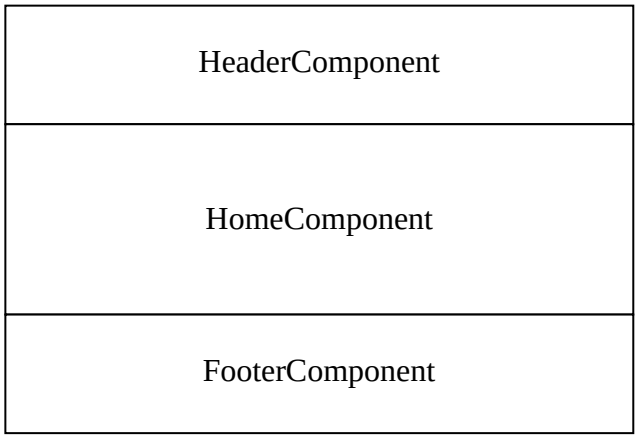

Рисунок 3.3. Головна сторінка

url адреса – http://localhost:4200/login

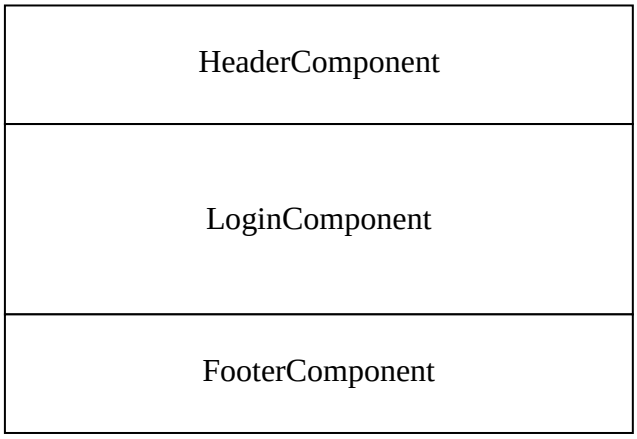

Рисунок 3.4. Сторінка входу в систему

url адреса – http://localhost:4200/register

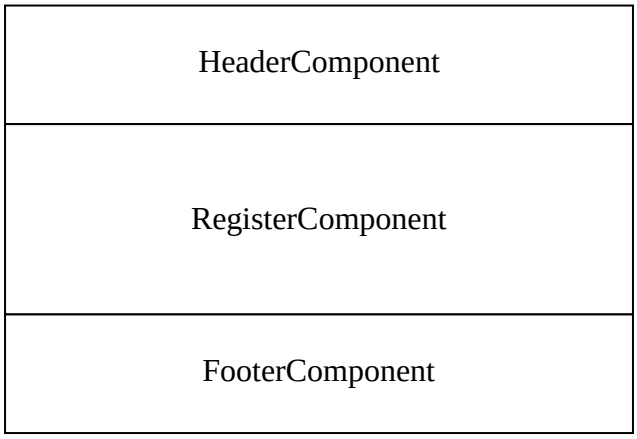

Рисунок 3.5. Сторінка реєстрації в системі

url адреса – http://localhost:4200/cabinet

Рисунок 3.6. Сторінка персонального кабінету

url адреса – http://localhost:4200/exercises

Рисунок 3.7. Сторінка зі списком вправ

url адреса – http://localhost:4200/exercises/ex1

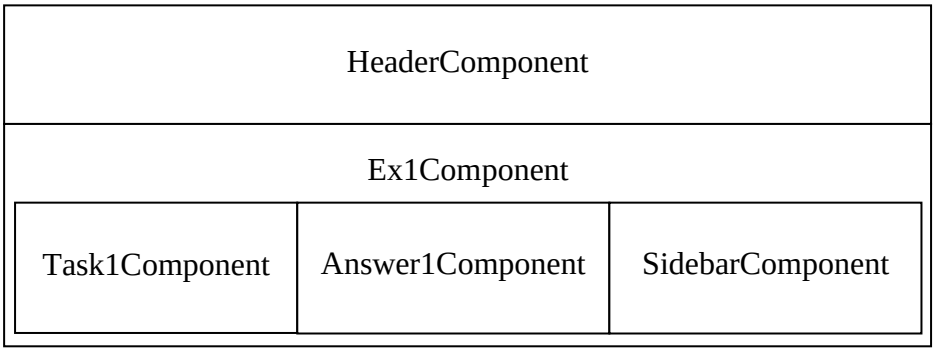

Рисунок 3.8. Сторінка вправи «Тренуй себе»

url адреса – http://localhost:4200/exercises/ex2

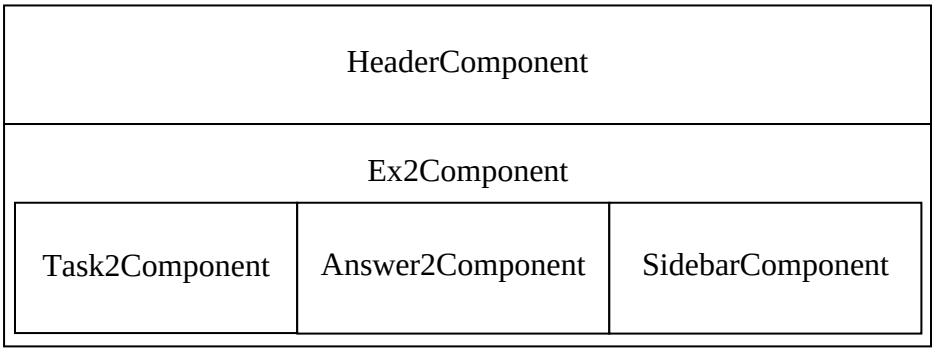

Рисунок 3.9. Сторінка вправи «Визнач результат»

url адреса – http://localhost:4200/exercises/ex3

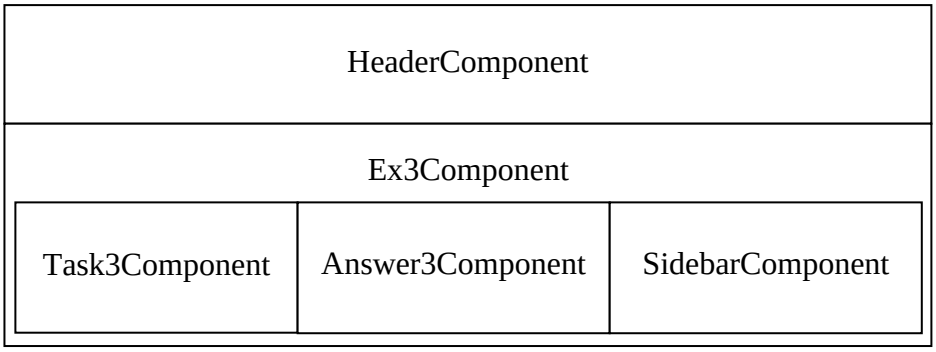

Рисунок 3.10. Сторінка вправи «Перевір знання»

url адреса – http://localhost:4200/additionally

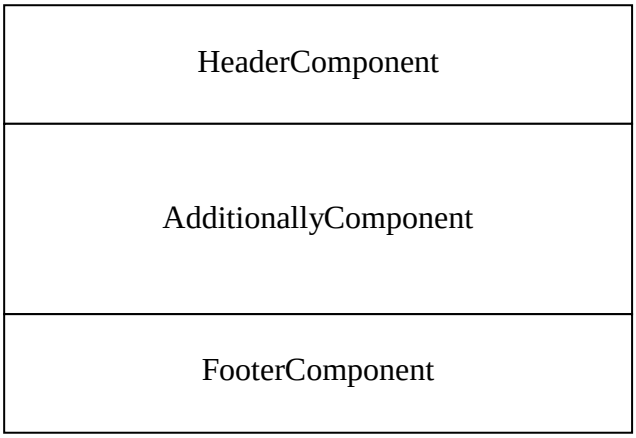

Рисунок 3.11. Сторінка з додатковою інформацією

3.3.2.2 Сервіси Angular

В ході роботи були створені наступні Angular сервіси:

– AuthenticationService – сервіс, що відповідає за авторизацію користувача в системі, завантаження його даних та вихід користувача з системи;
– RegisterService – сервіс, що відповідає за реєстрацію користувача в системі;
– UserService – сервіс, що відповідає за роботу з даними користувачів системи, оновлення або видалення даних користувача;
– TaskService – сервіс, що відповідає за отримання даних про доступні вправи та списки завдань для цих вправ;
– AlertService – сервіс, що відповідає за виведення службових повідомлень в разі виникнення помилок.

3.3.2.3 Специфікація функцій front-end частини

Функції Angular, залежно від компонентів наведені в таблиці 3.1.

Таблиця 3.1. Функції компонентів в Angular

Компонент	Функція	Опис функції
Login	constructor(private fb: FormBuilder, private route: ActivatedRoute, private router: Router, private alertService: AlertService, private authService: AuthenticationService)	Підключення сервісів роботи з формами, маршрутизацією, авторизацією
	ngOnInit()	Ініціалізація форми для входу в систему
	onSubmit()	Якщо користувач існує, то відбувається вхід до системи, в іншому випадку функція повідомляє про помилку.
Register	constructor(private fb: FormBuilder, private route: ActivatedRoute, private router: Router, private alertService: AlertService, private authService: AuthenticationService)	Підключення сервісів роботи з формами, маршрутизацією, авторизацією
	ngOnInit()	Ініціалізація форми реєстрації в системі
	onSubmit()	Перевіряє на правильність заповнення данних форми та реєструє в системі

Продовження таблиці 3.1

Компонент	Функція	Опис функції
Cabinet	resetAllProgress()	Скидання всього прогресу користувача до початкових значень
	resetEx1Progress()	Скидання прогресу вправи «Тренуй себе»
	resetEx2Progress()	Скидання прогресу вправи «Визнач результат»
	resetEx3Progress()	Скидання прогресу вправи «Перевір знання»
Header	constructor(private router: Router, private activatedRoute: ActivatedRoute, private authService: AuthenticationService)	Підключення сервісів авторизації та маршрутизації
	onLogout()	Вигрузити користувача з системи
ExercisesList	constructor(private taskService: TaskService)	Підключення сервісу завдань
	ngOnInit()	Обрахування всієї кількості завдань з бази даних, визначення поточного прогресу користувача.
Sidebar	onTaskSelect(levelId: number, taskId: number)	Відправлення в батьківський компонент даних про обране завдання зі списку

Продовження таблиці 3.1

Компонент	Функція	Опис функції
Ex1	constructor(private taskService: TaskService, private toastr: ToastrService)	Підключення сервісу завдань та сервісу показу повідомлень
	ngOnInit()	Назначити поточне завдання згідно з прогресом користувача при відкритті вправи
	onTaskChange(model: SelectedTaskModel)	Змінити завдання залежно від вибраного користувачем зі списку завдань
	onAnswerChecked()	Функція перевіряє, чи було це завдання останнім, і якщо ні, то перенаправляє на наступне завдання в списку та запам’ятовує новий прогрес користувача
Answer1	constructor(private toastr: ToastrService)	Підключення сервісу виведення повідомлень
	ngAfterViewInit()	Назначити налаштування до редактора коду Ace
	onChange(code)	Отримати код, написаний в редакторі при кожній зміні

Продовження таблиці 3.1

Компонент	Функція	Опис функції
	ex1Verification()	Перевірити на правильність виконання завдання, та в разі успіху відправити до батьківського компоненту результат, а також виводити повідомлення про правильність виконання
Ex2	constructor(private taskService: TaskService, private toastr: ToastrService)	Підключення сервісу завдань та сервісу показу повідомлень
	ngOnInit()	Назначити поточне завдання згідно з прогресом користувача при відкритті вправи
	onTaskChange(model: SelectedTaskModel)	Змінити завдання залежно від вибраного користувачем зі списку завдань
	onAnswerChecked()	Функція перевіряє, чи було це завдання останнім, і якщо ні, то перенаправляє на наступне завдання в списку та запам’ятовує новий прогрес користувача

Продовження таблиці 3.1

Компонент	Функція	Опис функції
Answer2	constructor(private toastr: ToastrService)	Підключення сервісу виведення повідомлень
	ex2Verification()	Перевірити на правильність виконання завдання, та в разі успіху відправити до батьківського компоненту результат, а також виводити повідомлення про правильність виконання
Ex3	constructor(private taskService: TaskService, private toastr: ToastrService)	Підключення сервісу завдань та сервісу показу повідомлень
	ngOnInit()	Назначити поточне завдання згідно з прогресом користувача при відкритті вправи
	onTaskChange(model: SelectedTaskModel)	Змінити завдання залежно від вибраного користувачем зі списку завдань

Продовження таблиці 3.1

Компонент	Функція	Опис функції
	onAnswerChecked()	Функція перевіряє, чи було це завдання останнім, і якщо ні, то перенаправляє на наступне завдання в списку та запам’ятовує новий прогрес користувача
Answer3	constructor(private toastr: ToastrService)	Підключення сервісу виведення повідомлень
	ex3Verification()	Перевірити на правильність виконання завдання, та в разі успіху відправити до батьківського компоненту результат, а також виводити повідомлення про правильність виконання

Функції Angular, залежно від сервісів наведені в таблиці 3.2.

Таблиця 3.2. Функції сервісів в Angular

Сервіс	Функція	Опис функції
Authentication	constructor(private http: HttpClient)	Підключення сервісу для роботи з http зверненнями

Продовження таблиці 3.2

Сервіс	Функція	Опис функції
	public get currentUserValue(): UserModel	Функція отримує дані поточного користувача системи
	public IsLogin(): boolean	Перевіряє, чи користувач війшов в систему
	authenticate(model: LoginModel)	Виконує вхід до системи користувача, завантажує його дані в локальне сховище браузера
	logout()	Виконує вихід користувача із системи
Register	constructor(private http: HttpClient)	Підключення сервісу для роботи з http зверненнями
	register(user: LoginModel)	Відправити дані про користувача на реєстрацію
User	constructor(private http: HttpClient)	Підключення сервісу для роботи з http зверненнями
	getAll()	Отримати дані про всіх користувачів
	getById(id: number)	Отримати дані про користувача з відомом id
	update(user: UserModel)	Обновити дані про користувача в базі даних
	delete(id: number)	Видалити користувача з бази даних

Продовження таблиці 3.2

Сервіс	Функція	Опис функції
Task	constructor(private http: HttpClient)	Підключення сервісу для роботи з http зверненнями
	getExercises()	Отримати список вправ з бази даних
	getEx1TaskList()	Отримати завдання для вправи «Тренуй себе»
	getEx2TaskList()	Отримати завдання для вправи «Визнач результат»
	getEx3TaskList()	Отримати завдання для вправи «Перевір знання»
Alert	constructor(private router: Router)	Підключення маршрутизації
	success(message: string, keepAfterNavigationChange = false)	Виводить повідомлення про успіх операції
	error(message: string, keepAfterNavigationChange = false)	Виводить повідомлення про помилку в виконанні операції
	getMessage()	Отримує повідомлення

3.3.3 Back-end частина

3.3.3.1 Маршрутизація

Маршрутизація визначає як сервер відповідає на клієнтський запит до певного url адресу.

Маршрути REST API Node.js наведені в таблиці 3.3

Таблиця 3.3. Маршрути REST API

Маршрут	Метод	Опис дій сервера
/api/user/list	GET	Відправляє клієнтові дані масива користувачів
/api/user/score	POST	Зберігає прогрес користувача
/api/auth/login	POST	Перевіряє чи існує такий користувач в масиві користувачів
/api/auth/logout	POST	Завершує поточну сесію користувача
/api/auth/register	POST	Добавляє нового користувача до масиву користувачів
/api/ex1/task/list	GET	Відправляє клієнтові список завдань із вправи «Тренуй себе»
/api/ex2/task/list	GET	Відправляє клієнтові список завдань із вправи «Визнач результат»
/api/ex3/task/list	GET	Відправляє клієнтові список завдань із вправи «Перевір знання»

3.3.3.2 Специфікація функцій back-end частини

Функції Node.js, залежно від модулів наведені в таблиці 3.4.

Таблиця 3.4. Функції Node.js

Модуль	Функція	Опис функції
authApi	userList(req, res)	Отримує коллекцію зі список користувачів
	login(req, res)	Перевіряє чи існує користувач з отриманим логіном та паролем в колекції користувачів
	logout(req, res)	Завершує поточну сесію користувача
	register(req, res)	Реєструє користувача в колекції користувачів
	saveScore(req, res)	Зберігає прогрес користувача
taskApi	exercise(req, res)	Отримує колекцію зі списком вправ
	ex1TaskList(req, res)	Отримує колекцію зі списком завдань вправи «Тренуй себе»
	ex2TaskList(req, res)	Отримує колекцію зі списком завдань вправи «Визнач результат»
	ex3TaskList(req, res)	Отримує колекцію зі списком завдань вправи «Перевір знання»

Висновок до розділу

В даному розділі була розглянута архітектура програмного забезпечення системи. Система поділена на два незалежні розділи, які взаємодіють за допомогою технології REST API для отримання та відправлення даних.

4 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

4.1 Керівництво користувача

Коли користувач заходить на веб-сайт він потрапляє на головну сторінку (рисунок 4.1).

Рисунок 4.1. Головна сторінка

Першим чином для подальшого користування можливостями веб-сайту користувач повинен війти в систему (рисунок 4.2) або зареєструватися (рисунок 4.3).

Рисунок 4.2. Сторінка авторизації

Рисунок 4.3. Сторінка реєстрації

Після реєстрації користувач може обрати одну з трьох доступних для виконання вправ (рисунок 4.4).

Рисунок 4.4. Сторінка вибору вправи

Вправа «Тренуй себе» містить в собі завдання та редактор для написання коду (рисунок 4.5).

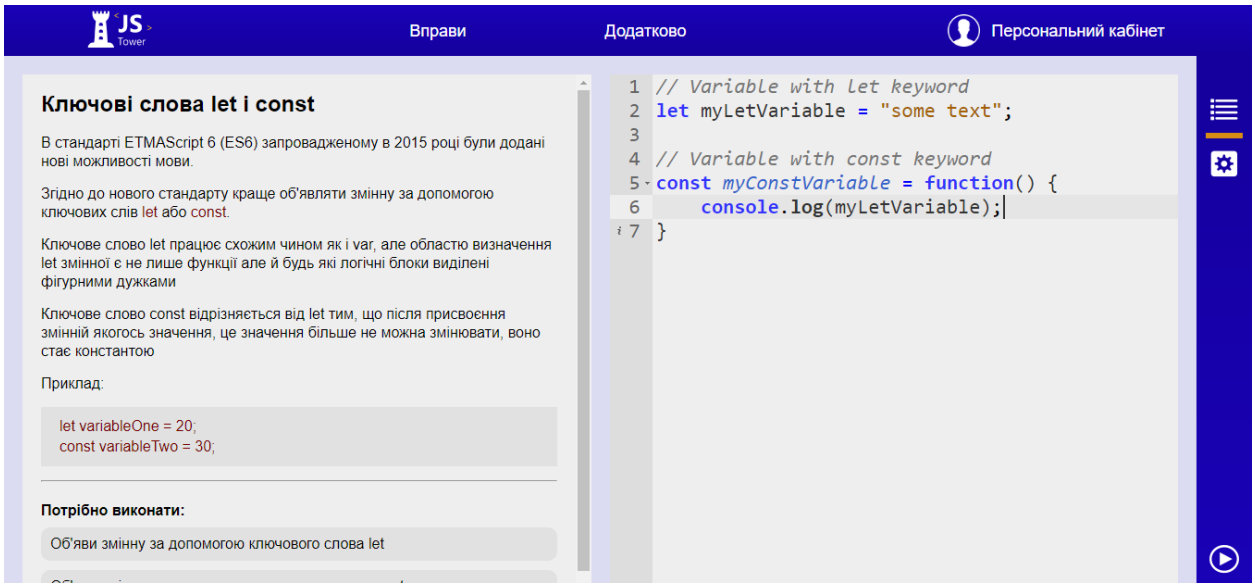

Рисунок 4.5. Сторінка вправи «Тренуй себе»

Вправа «Визнач результат» містить в собі завдання з кодом та текстове поле для введення відповіді (рисунок 4.6).

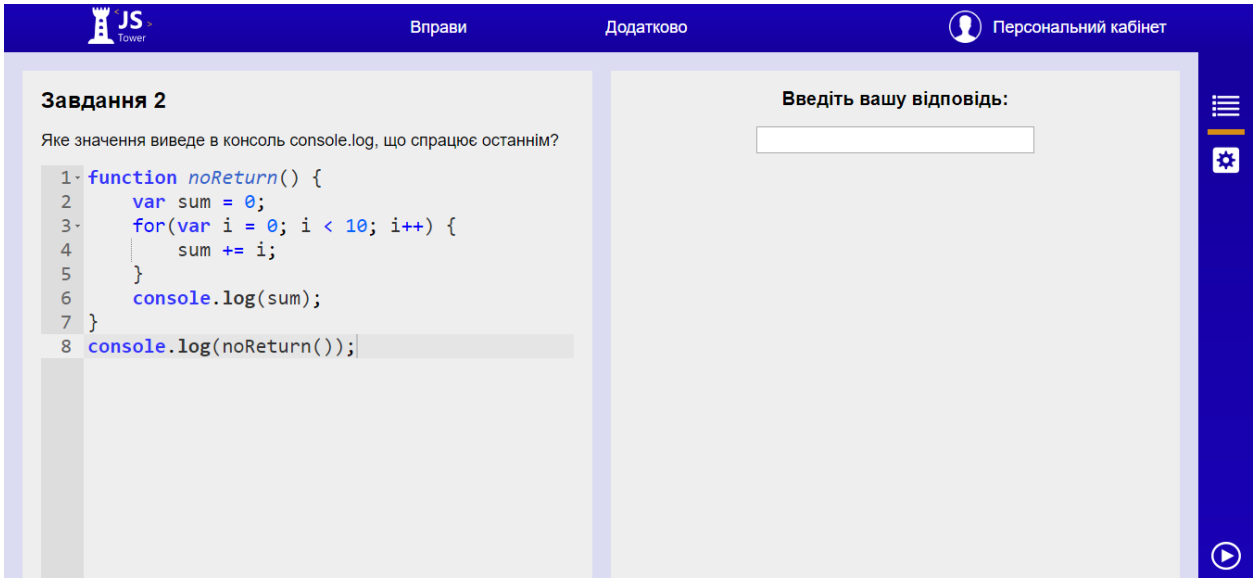

Рисунок 4.6. Сторінка вправи «Визнач результат»

Вправа «Перевір знання» містить в собі запитання та варіанти відповіді (рисунок 4.7).

Рисунок 4.7. Сторінка вправи «Перевір знання»

Також користувач може обрати завдання для вирішення зі списку завдань в боковій панелі (рисунок 4.8).

Рисунок 4.8. Список завдань

Якщо користувач зможе правильно виконати завдання, то система автоматично запам’ятає прогрес користувача та переведе його до наступного завдання (рисунок 4.9).

Рисунок 4.9. Список завдань

Якщо користувач допустить помилку, система його про це попередить (рисунок 4.10).

Рисунок 4.10. Список завдань

Користувач має змогу зайти до персонального кабінету, та виконати певні дії (рисунок 4.11)

Рисунок 4.11. Сторінка персонального кабінету

Якщо користувач бажає відвідати схожі ресурси, або знайти якусь інформацію по мові JavaScript, він може ввійти до вкладки «Додатково» та обрати те, що йому потрібно (рисунок 4.12).

Рисунок 4.12. Сторінка з додатковими ресурсами

4.2 Випробування програмного продукту

В таблицях 4.1 – 4.7 наведені випробування веб застосування.

Таблиця 4.1. Авторизація в системі

Мета тесту	**Перевірка функції авторизації в системі**
Початковий стан	Відкрити сторінку авторизації користувача
Вхідні дані	Корректний логін та пароль, що існують в колекції даних користувачів.
Схема проведення тесту	Користувач вводить свої дані для авторизації
Очікуваний результат	Студент зареєстрований, доступна функція "Вийти" під персональним кабінетом користувача
Стан після	Користувач авторизований в системі

Продовження таблиці 4.1

проведення випробувань	

Таблиця 4.2. Тест вправи «Тренуй себе»

Мета тесту	**Перевірка виконаного завдання вправи «Тренуй себе»**
Початковий стан	Відкрити сторінку вправи «Тренуй себе»
Вхідні дані	Завдання вправи «Тренуй себе»
Схема проведення тесту	Користувач пише код, що вирішує поставлене завдання
Очікуваний результат	Система перевірить код на правильність. Якщо завдання виконане вірно, система покаже користувачу наступне завдання, в іншому разі укаже, що в рішенні завдання є помилка.
Стан після проведення випробувань	Користувач може вирішувати наступне завдання.

Таблиця 4.3. Тест вправи «Визнач результат»

Мета тесту	**Перевірка виконаного завдання вправи «Визнач результат»**
Початковий стан	Відкрити сторінку вправи «Визнач результат»
Вхідні дані	Завдання вправи «Визнач результат»
Схема проведення тесту	Користувач вводить відповідь до завдання в текстове поле
Очікуваний результат	Система перевірить відповідь на правильність. Якщо завдання виконане вірно, система покаже користувачу наступне завдання, в іншому разі укаже, що в рішенні завдання є помилка.

Продовження таблиці 4.3

Стан після проведення випробувань	Користувач може вирішувати наступне завдання.

Таблиця 4.4. Тест вправи «Перевір знання»

Мета тесту	**Перевірка виконаного завдання вправи «Перевір знання»**
Початковий стан	Відкрити сторінку вправи «Перевір знання»
Вхідні дані	Завдання вправи «Перевір знання»
Схема проведення тесту	Користувач вибирає одну з запропонованих відповідей до завдання
Очікуваний результат	Система перевірить відповідь на правильність. Якщо завдання виконане вірно, система покаже користувачу наступне завдання, в іншому разі укаже, що в рішенні завдання є помилка.
Стан після проведення випробувань	Користувач може вирішувати наступне завдання.

Таблиця 4.5. Тест вибору завдання із списку завдань в боковій панелі

Мета тесту	**Перевірка правильності роботи списку завдань в боковій панелі**
Початковий стан	Відкрити сторінку одної з доступних вправ
Вхідні дані	Список всіх завдань до оброної вправи
Схема проведення тесту	Користувач відкриває список завдань в боковій панелі та обирає одне з запропонованих завдань
Очікуваний результат	Система перенаправить користувача на обране завдання
Стан після	Користувач може вирішувати обране завдання.

Продовження таблиці 4.5

проведення випробувань	

Таблиця 4.6. Тест на запам’ятовування прогресу користувача

Мета тесту	**Перевірка на правильність запам’ятовування прогресу користувача при виконанні вправи**
Початковий стан	Відкрити сторінку одної з доступних вправ
Вхідні дані	Завдання обраної вправи
Схема проведення тесту	Користувач вирішує поставлене завдання та натискає кнопку перевірки. В разі успішного виконання завдання система запам’ятає про вирішення цього завдання користувачем
Очікуваний результат	Збережений прогрес користувача
Стан після проведення випробувань	Система працює правильно, прогрес користувача зберігається

Таблиця 4.7. Тест на скидання прогресу користувача

Мета тесту	**Перевірка на правильність роботи функції скидання прогресу користувача**
Початковий стан	Відкрити сторінку особистого кабінету
Вхідні дані	Дані про прогрес користувача
Схема проведення тесту	Користувач натискає на кнопку скидання прогресу одної з вправ.
Очікуваний результат	Обнулений прогрес користувача
Стан після проведення випробувань	Система працює правильно, прогрес користувача обнуляється

Таблиця 4.8. Тест на відкриття останнього невиконаного завдання

Мета тесту	**Перевірка на перехід до останнього невиконаного завдання при відкритті вправи**
Початковий стан	Користувач виконав певну кількість завдань з вправи
Вхідні дані	Дані про прогрес користувача
Схема проведення тесту	Користувач виходить з вправи, а потім заходить знову.
Очікуваний результат	Перенаправлення до останнього завдання, на якому зупинився користувач
Стан після проведення випробувань	Відкрите останнє невиконане завдання користувача

4.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій інтерактивної системи підтримки вивчення мови програмування JavaScript вимогам технічного завдання.

4.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

- ГОСТ 34.603−92. Інформаційна технологія. Види випробувань автоматизованих систем;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

4.2.3 Результати випробувань

Всі тести виконані успішно. Була перевірена вся функціональність інтерактивної системи підтримки вивчення мови програмування JavaScript

Висновок до розділу

В даному розділі було наведено керівництво користувача, доступні функції системи та методи взаємодії з ними. Були описані тести на правильність роботи системи та відповідні очікувані результати. Всі випробування було пройдено успішно.

ЗАГАЛЬНІ ВИСНОВКИ

В ході виконання дипломного проекту були детально розглянуті способи та можливості вивчення мови програмування JavaScript. Створено систему, що задовільняє оприділеним критеріям для вивчення мови JavaScript.

Було розглянуто процес діяльності, способи створення користувацького інтерфейсу, взято до уваги зручність в користуванні веб-застосуванням.

На основі розглянутих аналогів було визначено, які переваги та недоліки в них наявні, та як можна покращити взаємодію користувача з системою.

Для розробки програмного забезпечення був використаний фреймворк Angular та Node.js. Використовувалося середовище розробки Visual Studio Code. Була детально розглянута архітектура системи.

Для збереження даних була використана база даних LokiJS, та продумана модель структури даних, що будуть задовільняти вимогам системи.

Наведена детальна інструкція користувача по використанню системи, описана методика випробувань.

В результаті виконання дипломного проекту було отримано досвід розробки повноцінного веб-застосування з приміненням технологій Angular та Node.js.

ПЕРЕЛІК ПОСИЛАНЬ

1) Node.JS [Електронний ресурс] – Режим доступу URL: https://nodejs.org/
2) LokiJS [Електронний ресурс] – Режим доступу URL: http://lokijs.org/
3) Git [Електронний ресурс] – Режим доступу URL: https://git-scm.com/
4) Github [Електронний ресурс] – Режим доступу URL: https://github.com/
5) Angular [Електронний ресурс] – Режим доступу URL: https://angular.io/
6) Visual Studio Code [Електронний ресурс] – Режим доступу URL: https://code.visualstudio.com/
7) JavaScript [Електронний ресурс] – Режим доступу URL: https://learn.javascript.ru/
8) TypeScript [Електронний ресурс] – Режим доступу URL: https://www.typescriptlang.org/
9) SASS [Електронний ресурс] – Режим доступу URL: https://sass-lang.com/
10) Ace [Електронний ресурс] – Режим доступу URL: https://ace.c9.io/
11) Express [Електронний ресурс] – Режим доступу URL: http://expressjs.com/
12) Figma [Електронний ресурс] – Режим доступу URL: https://www.figma.com/

Додаток А

Тексти програмного коду

Інтерактивна система підтримки вивчення мови програмування JavaScript

(Найменування програми (документа))

DVD-R

(Вид носія даних)

49 арк, 732 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

Лістинги файлів front-end частини

Лістинг файлу «package.json»:

```
{
  "name": "js-tower",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve --proxy-config src/proxy.conf.json",
    "server": "nodemon ../server/main.js",
    "all": "npm-run-all -p start server",
    "build": "ng build --output-path=dist && npm run server:build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "nodemonConfig": {
    "watch": [
      "../server/app"
    ],
    "ext": "js",
    "ignore": [
      "../server/js-tower.json"
    ],
    "delay": "3"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~7.2.0",
    "@angular/common": "~7.2.0",
    "@angular/compiler": "~7.2.0",
    "@angular/core": "~7.2.0",
    "@angular/forms": "~7.2.0",
    "@angular/platform-browser": "~7.2.0",
    "@angular/platform-browser-dynamic": "~7.2.0",
    "@angular/router": "~7.2.0",
    "ace-code-editor": "^1.2.3",
    "core-js": "^2.5.4",
    "ng2-ace-editor": "^0.3.9",
    "ngx-toastr": "^10.0.4",
    "nodemon": "^1.19.0",
    "npm-run-all": "^4.1.5",
    "popper.js": "^1.15.0",
    "rxjs": "~6.3.3",
    "tslib": "^1.9.0",
    "zone.js": "~0.8.26"
```

```
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.13.0",
    "@angular/cli": "~7.3.1",
    "@angular/compiler-cli": "~7.2.0",
    "@angular/language-service": "~7.2.0",
    "@types/jasmine": "~2.8.8",
    "@types/jasminewd2": "~2.0.3",
    "@types/jquery": "^3.3.29",
    "@types/node": "~8.9.4",
    "codelyzer": "~4.5.0",
    "jasmine-core": "~2.99.1",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "^4.1.0",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "~2.0.1",
    "karma-jasmine": "~1.1.2",
    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.4.0",
    "ts-node": "~7.0.0",
    "tslint": "~5.11.0",
    "typescript": "~3.2.2"
  }
}
```

Лістинг файлу «tsconfig.json»:

```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "module": "es2015",
    "moduleResolution": "node",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "importHelpers": true,
    "target": "es5",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2018",
      "dom"
```

```
    ],
    "paths": {
      "@/*": [
        "src/app/*"
      ],
      "@env/*": [
        "src/environments/*"
      ]
    }
  }
}
```

Лістинг файлу «app.module.ts»:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';

import { AceEditorModule } from 'ng2-ace-editor';
import { ToastrModule } from 'ngx-toastr';

import { JwtInterceptor, ErrorInterceptor, BodyTranspilator } from '@/helpers';

import { AppRoutingModule } from '@/app-routing.module';

import { AppComponent } from '@/app.component';
import { HomeComponent } from '@/home';
import { HeaderComponent } from '@/header';
import { FooterComponent } from '@/footer';
import { LoginComponent } from '@/login';
import { RegisterComponent } from '@/register';

import { ExercisesAllComponent, ExercisesListComponent, SidebarComponent } from
'@/exercises';
import { Ex1Component, Task1Component, Answer1Component } from '@/exercises/ex1';
import { Ex2Component, Task2Component, Answer2Component } from '@/exercises/ex2';
import { Ex3Component, Task3Component, Answer3Component } from '@/exercises/ex3';

import { AlertComponent } from '@/_shared/alert.component';
import { CabinetComponent } from '@/cabinet';
import {BrowserAnimationsModule} from '@angular/platform-browser/animations';
import { AdditionallyComponent } from './additionally/additionally.component';

@NgModule({
  declarations: [
```

```
    AppComponent,
    HomeComponent,
    HeaderComponent,
    Ex1Component,
    Task1Component,
    Answer1Component,
    ExercisesAllComponent,
    FooterComponent,
    SidebarComponent,
    ExercisesListComponent,
    LoginComponent,
    RegisterComponent,
    AlertComponent,
    CabinetComponent,
    Ex2Component,
    Ex3Component,
    Task2Component,
    Answer2Component,
    Answer3Component,
    Task3Component,
    AdditionallyComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    ToastrModule.forRoot({
      timeOut: 5000,
      positionClass: 'toast-top-right'
    }),
    AppRoutingModule,
    ReactiveFormsModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    AceEditorModule
  ],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: ErrorInterceptor, multi: true },
    BodyTranspilator
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Лістинг файлу «app-routing.module.ts»:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home';
import { ExercisesAllComponent, ExercisesListComponent } from './exercises';
import { Ex1Component } from './exercises/ex1';
import { Ex2Component } from './exercises/ex2';
import { Ex3Component } from './exercises/ex3';
import { LoginComponent } from './login';
import { RegisterComponent } from './register';
import { AuthGuard } from './_guards';
import { CabinetComponent } from './cabinet';
import { AdditionallyComponent } from './additionally';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: 'cabinet', component: CabinetComponent, canActivate: [AuthGuard] },
  { path: 'additionally', component: AdditionallyComponent },
  {
    path: 'exercises',
    component: ExercisesAllComponent,
    children: [
      { path: '', component: ExercisesListComponent },
      { path: 'ex1', component: Ex1Component, canActivate: [AuthGuard] },
      { path: 'ex2', component: Ex2Component, canActivate: [AuthGuard] },
      { path: 'ex3', component: Ex3Component, canActivate: [AuthGuard] }
    ]
  },
  {path: '**', redirectTo: '' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Лістинг файлу «authentication.service.ts»:

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { BehaviorSubject, Observable } from 'rxjs';
import { map } from 'rxjs/operators';
```

```
import { UserModel, LoginModel } from '@/models';
import { environment } from '@env/environment';
import { BodyTranspilator } from '@/helpers/bodyTranspilator';

@Injectable({
  providedIn: 'root'
})

export class AuthenticationService {
  headers: HttpHeaders;
  domainURL = environment.apiUrl;
  isLogin = false;
  private currentUserSubject: BehaviorSubject<UserModel>;
  public currentUser: Observable<UserModel>;

  constructor(private http: HttpClient, private transpilator: BodyTranspilator) {
    this.headers = new HttpHeaders({ 'Content-Type': 'application/x-www-form-urlencoded'
});
    this.currentUserSubject = new
BehaviorSubject<UserModel>(JSON.parse(localStorage.getItem('currentUser')));
    this.currentUser = this.currentUserSubject.asObservable();
  }

  public get currentUserValue(): UserModel {
    return this.currentUserSubject.value;
  }

  public IsLogin(): boolean {
    return this.isLogin;
  }

  authenticate(model: LoginModel) {
    const body = this.transpilator.transform(model);
    return this.http.post<any>(`${this.domainURL}/api/auth/login`, body, {headers:
this.headers})
      .pipe(map(res => {
        const user = res.data;
        // login successful if there's a jwt token in the response
        if (user && user.token) {
          // store user details and jwt token in local storage to keep user logged in between page
refreshes
          localStorage.setItem('currentUser', JSON.stringify(user));
          this.isLogin = true;
          this.currentUserSubject.next(user);
        }
        return user;
```

```
    }));
  }

  logout() {
    // remove user from local storage to log user out
    localStorage.removeItem('currentUser');
    this.isLogin = false;
    this.currentUserSubject.next(null);
  }
}
```

Лістинг файлу «task.service.ts»:

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';
import { Observable } from 'rxjs';
import { map } from 'rxjs/operators';

import { ExerciseModel, TaskLevelModel } from '@/models';
import { environment } from '@env/environment';

@Injectable({
  providedIn: 'root'
})

export class TaskService {
  headers: HttpHeaders;
  domainURL = environment.apiUrl;

  constructor( private http: HttpClient) {
    this.headers = new HttpHeaders({ 'Content-Type': 'application/json' });
  }

  getEx1TaskList(): Observable<any> {
    const reqOpts = { headers: this.headers };
    return this.http.get(`${this.domainURL}/api/ex1/task/list`, reqOpts).pipe(map(data =>
data['data']));
  }

  getEx2TaskList(): Observable<any> {
    const reqOpts = { headers: this.headers };
    return this.http.get(`${this.domainURL}/api/ex2/task/list`, reqOpts).pipe(map(data =>
data['data']));
  }

  getEx3TaskList(): Observable<any> {
```

```
    const reqOpts = { headers: this.headers };
    return this.http.get(`${this.domainURL}/api/ex3/task/list`, reqOpts).pipe(map(data =>
data['data']));
  }

  getExercises(): Observable<any> {
    const reqOpts = {
      params: new HttpParams(),
      headers: this.headers
    };
    reqOpts.params = reqOpts.params.set('taskId', id.toString());
    return this.http.get(`${this.domainURL}/api/task/exercise`, reqOpts).pipe(map(data => {
      return data['data'];
    }));
  }
}
```

Лістинг файлу «user.service.ts»:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

import {LoginModel, UserModel} from '@/models';
import { environment } from '@env/environment';

@Injectable({
  providedIn: 'root'
})
export class UserService {
  constructor(private http: HttpClient) { }

  getAll() {
    return this.http.get<UserModel[]>(`${environment.apiUrl}/users`);
  }

  getById(id: number) {
    return this.http.get(`${environment.apiUrl}/users/${id}`);
  }

  register(user: LoginModel) {
    return this.http.post(`${environment.apiUrl}/users/register`, user);
  }

  update(user: UserModel) {
    return this.http.put(`${environment.apiUrl}/users/${user.id}`, user);
  }
```

```
  delete(id: number) {
    return this.http.delete(`${environment.apiUrl}/users/${id}`);
  }
}
```

Лістинг файлу «app.guard.ts»:

```
import { Injectable } from '@angular/core';
import { Router, CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from
'@angular/router';

// @ts-ignore
import { AuthenticationService } from '../_services';

@Injectable({ providedIn: 'root' })
export class AuthGuard implements CanActivate {
  constructor(
    private router: Router,
    private authenticationService: AuthenticationService
  ) {}

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    const currentUser = this.authenticationService.currentUserValue;
    if (currentUser) {
      // authorised so return true
      return true;
    }

    // not logged in so redirect to login page with the return url
    this.router.navigate(['/login'], { queryParams: { returnUrl: state.url }});
    return false;
  }
}
```

Лістинг файлу «app.component.html»:

```
<!-- Site header -->
<app-header></app-header>
<app-alert></app-alert>
<!-- Router -->
<router-outlet></router-outlet>
```

Лістинг файлу «app.component.ts»:

```
import { Component } from '@angular/core';

@Component({
```

```
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
}
```

Лістинг файлу «login.component.html»:

```
<section class="auth-section">
  <div class="container">
    <h2>Вхід для зареєстрованих користувачів</h2>
    <form [formGroup]="loginForm" (ngSubmit)="onSubmit()" novalidate>
      <div class="form-group">
        <label for="login">Логін <span>*</span></label>
        <input type="text" id="login" formControlName="login" class="form-control"
required>
        <div *ngIf="submitted && f.login.errors" class="invalid-feedback">
          <div class="error" *ngIf="f.login.errors.required">Введіть корректний
логін</div>
        </div>
      </div>
      <div class="form-group">
        <label for="password">Пароль <span>*</span></label>
        <input type="password" id="password" formControlName="password"
class="form-control" required>
        <div *ngIf="submitted && f.password.errors" class="invalid-feedback">
          <div class="error" *ngIf="f.password.errors.required">Введіть корректний
пароль</div>
        </div>
      </div>
      <div class="form-buttons">
        <button type="submit" [disabled]="loading">Війти</button>
        <img *ngIf="loading" class="pl-3"
src="
CgpKSkiH/C05FVFNDQVBFMi4wAwEAAAAh/hpDcmVhdGVkIHdpdGggYWpheGxvY
WQuaW5mbwAh+QQJCgAAACwAAAAAEAAQAAADMwi63P4wyklrE2MIOggZnAdO
mGYJRbExwroUmcG2LmDEwnHQLVsYOd2mBzkYDAdKa+dIAAAh+QQJCgAAACw
AAAAAEAAQAAADNAi63P5OjCEgG4QMu7DmikRxQlFUYDEZIGBMRVsaqHwctXXf
7WEYB4Ag1xjihkMZsiUkKhIAIfkECQoAAAAsAAAAABAAEAAAAzYIujIjK8pByJDM
lFYvBoVjHA70GU7xSUJhmKtwHPAKzLO9HMaoKwJZ7Rf8AYPDDzKpZBqfvwQAIfk
ECQoAAAAsAAAAABAAEAAAAzMIumIlK8oyhpHsnFZfhYumCYUhDAQxRIdhHBGq
RoKw0R8DYlJd8z0fMDgsGo/IpHI5TAAAIfkECQoAAAAsAAAAABAAEAAAAzIIunIn
K0rnZBTwGPNMgQwmdsNgXGJUlIWEuR5oWUIpz8pAEAMe6TwfwyYsGo/IpFKSAA
Ah+QQJCgAAACwAAAAAEAAQAAADMwi6IMKQORfjdOe82p4wGccc4CEuQradyles
ojEMBgsUc2G7sDX3lQGBMLAJibufbSlKAAAh+QQJCgAAACwAAAAAEAAQAAAD
```

```
Mgi63P7wCRHZnFVdmgHu2nFwlWCI3WGc3TSWhUFGxTAUkGCbtgENBMJAEJsxg
MLWzpEAACH5BAkKAAAALAAAAAAQABAAAAMyCLrc/jDKSatlQtScKdceCAjDII
7HcQ4EMTCpyrCuUBjCYRgHVtqlAiB1YhiCnlsRkAAAOwAAAAAAAAAAA==" />
      <a routerLink="/register">Зареєструватися</a>
    </div>
  </form>
 </div>
</section>
<app-footer></app-footer>
```

Лістинг файлу «login.component.ts»:

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router, ActivatedRoute } from '@angular/router';
import { first } from 'rxjs/operators';

import { AlertService, AuthenticationService } from '@/_services';
import { LoginModel } from '@/models';

@Component({
 selector: 'app-login',
 templateUrl: './login.component.html',
 styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {
 loginForm: FormGroup;
 loading = false;
 submitted = false;
 returnUrl: string;

 constructor(private fb: FormBuilder,
          private route: ActivatedRoute,
          private router: Router,
          private alertService: AlertService,
          private authService: AuthenticationService) {
  if (this.authService.currentUserValue) {
   this.router.navigate(['/']);
  }
 }

 ngOnInit() {
  this.loginForm = this.fb.group({
   login: ['', Validators.required],
   password: ['', Validators.required]
  });
```

```
    this.returnUrl = this.route.snapshot.queryParams['returnUrl'] || '/';
  }

  get f() { return this.loginForm.controls; }

  onSubmit() {
    this.submitted = true;
    if (this.loginForm.invalid) {
      return;
    }

    this.loading = true;
    const model = new LoginModel(this.f.login.value, this.f.password.value);
    this.authService.authenticate(model).pipe(first()).subscribe(
      data => {
        this.router.navigate(['/exercises']);
      },
      error => {
        this.alertService.error(error);
        this.loading = false;
      }
    );
  }
}
```

Лістинг файлу «register.component.html»:

```
<section class="auth-section">
  <div class="container">
    <h2>Реєстрація на сайті</h2>
    <p class="info-text">Зареєструвавшись на сайті ви зможете зберегти свій
прогрес!</p>
    <form [formGroup]="registerForm" (ngSubmit)="onSubmit()" novalidate>
      <div class="form-group">
        <label for="login">Логін <span>*</span></label>
        <input type="text" id="login" formControlName="login" required>
        <div *ngIf="submitted && f.login.errors" class="invalid-feedback">
          <div class="error" *ngIf="f.login.errors.required">Введіть коректний
логін</div>
        </div>
      </div>
      <div class="form-group">
        <label for="password">Пароль <span>*</span></label>
        <input type="password" id="password" formControlName="password" required>
        <div *ngIf="submitted && f.password.errors" class="invalid-feedback">
```

```
        <div class="error" *ngIf="f.password.errors.required">Введіть корректний
пароль</div>
      </div>
    </div>
    <div class="form-group">
      <label for="passwordRepeat">Підтвердження пароля <span>*</span></label>
      <input type="password" id="passwordRepeat"
formControlName="passwordRepeat" required>
      <div *ngIf="submitted && f.passwordRepeat.errors" class="invalid-feedback">
        <div class="error" *ngIf="f.passwordRepeat.errors.required">Введіть
корректний пароль</div>
      </div>
    </div>
    <div class="form-buttons">
      <button type="submit" [disabled]="loading">Зареєструватися</button>
      <img *ngIf="loading" class="pl-3"
src="
CgpKSkiH/C05FVFNDQVBFMi4wAwEAAAAh/hpDcmVhdGVkIHdpdGggYWpheGxvY
WQuaW5mbwAh+QQJCgAAACwAAAAAEAAQAAADMwi63P4wyklrE2MIOggZnAdO
mGYJRbExwroUmcG2LmDEwnHQLVsYOd2mBzkYDAdKa+dIAAAh+QQJCgAAACw
AAAAAEAAQAAADNAi63P5OjCEgG4QMu7DmikRxQlFUYDEZIGBMRVsaqHwctXXf
7WEYB4Ag1xjihkMZsiUkKhIAIfkECQoAAAAsAAAAABAAEAAAAzYIujIjK8pByJDM
lFYvBoVjHA70GU7xSUJhmKtwHPAKzLO9HMaoKwJZ7Rf8AYPDDzKpZBqfvwQAIfk
ECQoAAAAsAAAAABAAEAAAAzMIumIlK8oyhpHsnFZfhYumCYUhDAQxRIdhHBGq
RoKw0R8DYlJd8z0fMDgsGo/IpHI5TAAAIfkECQoAAAAsAAAAABAAEAAAAzIIunIn
K0rnZBTwGPNMgQwmdsNgXGJUlIWEuR5oWUIpz8pAEAMe6TwfwyYsGo/IpFKSAA
Ah+QQJCgAAACwAAAAAEAAQAAADMwi6IMKQORfjdOe82p4wGccc4CEuQradyles
ojEMBgsUc2G7sDX3lQGBMLAJibufbSlKAAAh+QQJCgAAACwAAAAAEAAQAAAD
Mgi63P7wCRHZnFVdmgHu2nFwlWCI3WGc3TSWhUFGxTAUkGCbtgENBMJAEJsxg
MLWzpEAACH5BAkKAAAALAAAAAAQABAAAAMyCLrc/jDKSatlQtScKdceCAjDII
7HcQ4EMTCpyrCuUBjCYRgHVtqlAiB1YhiCnlsRkAAAOwAAAAAAAAAAA==" />
      <a routerLink="/login">Назад</a>
    </div>
  </form>
 </div>
</section>
<app-footer></app-footer>
```

Лістинг файлу «register.component.ts»:

```
import { Component, OnInit } from '@angular/core';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';

import {AlertService, AuthenticationService, UserService} from '../_services';
import {Router} from '@angular/router';
import {first} from 'rxjs/operators';
```

```
import {LoginModel, UserModel} from '../models';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
})
export class RegisterComponent implements OnInit {
  registerForm: FormGroup;
  loading = false;
  submitted = false;

  constructor(
    private formBuilder: FormBuilder,
    private router: Router,
    private authService: AuthenticationService,
    private userService: UserService,
    private alertService: AlertService
  ) {
    // redirect to home if already logged in
    if (this.authService.currentUserValue) {
      this.router.navigate(['/']);
    }
  }

  ngOnInit() {
    // register form with FormBuilder
    this.registerForm = this.formBuilder.group({
      login: ['', Validators.required],
      password: ['', Validators.required],
      passwordRepeat: ['', Validators.required]
    });
  }

  get f() { return this.registerForm.controls; }

  onSubmit() {
    this.submitted = true;
    if (this.registerForm.invalid) {
      return;
    }

    this.loading = true;
    const user = new LoginModel(this.f.login.value, this.f.password.value);
    this.userService.register(user).pipe(first())
      .subscribe(
```

```
      data => {
        this.alertService.success('Registration successful', true);
        this.router.navigate(['/login']);
      },
      error => {
        this.alertService.error(error);
        this.loading = false;
      });
  }
}
```

Лістинг файлу «header.component.html»:

```
<header id="header" class="header">
  <div class="container">
    <div class="navigation">
      <a routerLink="" class="logo">
        <img src="assets/img/logo.png" alt="JS Tower">
      </a>
      <nav class="nav-menu">
        <ul class="menu">
          <li><a routerLink="/exercises">Вправи</a></li>
          <li><a routerLink="/additionally">Додатково</a></li>
        </ul>
      </nav>
      <!-- Personal cabinet -->
      <div class="cabinet">
        <div class="cabinet__inner" *ngIf="auth">
          <a routerLink="/cabinet">
            <img src="assets/img/user-replacement.png" alt="User photo" class="photo">
            <span>Персональний кабінет</span>
          </a>
          <div class="cabinet__dropdown">
            <a href="/login" class="logout" (click)="onLogout()">
              <img src="assets/img/sign-in.svg" alt="" class="sign-in">
              <span>Вийти</span>
            </a>
          </div>
        </div>
        <a routerLink="/login" *ngIf="!auth">
          <img src="assets/img/sign-in.svg" alt="" class="sign-in">
          <span>Війти</span>
        </a>
      </div>
    </div>
  </div>
```

```
</header>
```

Лістинг файлу «header.component.ts»:

```
import { Component, OnInit } from '@angular/core';
import {AuthenticationService} from '@/_services';
import { Router, ActivatedRoute} from '@angular/router';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.scss']
})
export class HeaderComponent implements OnInit {
  auth: boolean = false;
  isCabinet: boolean = false;

  constructor(
    private router: Router,
    private activatedRoute: ActivatedRoute,
    private authService: AuthenticationService
  ) {
    const auth = localStorage.getItem('currentUser');
    if(auth) this.auth = true;
  }

  ngOnInit() {
    if ( this.router.isActive('/cabinet', true) ) {
      this.isCabinet = true;
    }
  }

  onLogout() {
    this.authService.logout();
    this.router.navigate(['/']);
  }
}
```

Лістинг файлу «footer.component.html»:

```
<footer id="footer" class="footer">
  <div class="container">
    <p class="author">Робота студента Даниленка Івана</p>
  </div>
</footer>
```

Лістинг файлу «exercises-all.component.html»:

```
<router-outlet></router-outlet>
```

Лістинг файлу «exercises-list.component.html»:

```
<section id="exercises" class="exercises">
 <div class="container">
  <h1 class="exercises-title">Вправи для вивчення JavaScript</h1>
  <!-- exercises-list -->
  <ul class="exercises-list">
   <!-- ex1 -->
   <li class="purple">
    <a routerLink="/exercises/ex1" class="icon-comet">
     <div class="exercises-list__content">
      <h2>Тренуй себе</h2>
      <p>Вправа розрахована на опрацювання конструкцій мови на практиці</p>
     </div>
     <div class="progress">
      <!-- Progress ex 1 -->
      <span>Прогрес {{ ex1Current }}/{{ ex1Total }}</span>
     </div>
    </a>
   </li>
   <!-- ex2 -->
   <li class="blue">
    <a routerLink="/exercises/ex2" class="icon-clever">
     <div class="exercises-list__content">
      <h2>Визнач результат</h2>
      <p>Докажи, що можеш зрозуміти як працює будь-який код, яким би хитрим
він не був</p>
     </div>
     <div class="progress">
      <!-- Progress ex 2 -->
      <span>Прогрес {{ ex2Current }}/{{ ex2Total }}</span>
     </div>
    </a>
   </li>
   <!-- ex3 -->
   <li class="purple-light">
    <a routerLink="/exercises/ex3" class="icon-around-the-world">
     <div class="exercises-list__content">
      <h2>Перевір знання</h2>
      <p>Перевір свої теоретичні знання, відповівши на всі поставлені
запитання</p>
     </div>
     <div class="progress">
      <!-- Progress ex 3 -->
```

```
          <span>Прогресс {{ ex3Current }}/{{ ex3Total }}</span>
        </div>
      </a>
    </li>
  </ul>
 </div>
</section>
<app-footer></app-footer>
```

Лістинг файлу «exercises-list.component.ts»:

```
import { Component, OnInit } from '@angular/core';
import { TaskService } from '@/_services/task.service';

@Component({
 selector: 'app-exercises-list',
 templateUrl: './exercises-list.component.html',
 styleUrls: ['./exercises-list.component.scss']
})
export class ExercisesListComponent implements OnInit {
 ex1Current: number;
 ex2Current: number;
 ex3Current: number;
 ex1Total: number = 10;
 ex2Total: number = 10;
 ex3Total: number = 10;

 constructor(private taskService: TaskService) { }

 ngOnInit() {
  // get total number of all tasks for EX 1
  this.taskService.getEx1TaskList().subscribe(data => {
   for (let i = 0; i < data.length; i++) {
    this.ex1Total += data[i].tasks.length;
   }
  });
  this.taskService.getEx2TaskList().subscribe(data => {
   for (let i = 0; i < data.length; i++) {
    this.ex2Total += data[i].tasks.length;
   }
  });
  this.taskService.getEx3TaskList().subscribe(data => {
   for (let i = 0; i < data.length; i++) {
    this.ex3Total += data[i].tasks.length;
   }
  });
```

```
    // get progress of curent user
    this.ex1Current = JSON.parse(localStorage.getItem('currentUser')).progress.ex1Score;
    this.ex2Current = JSON.parse(localStorage.getItem('currentUser')).progress.ex2Score;
    this.ex3Current = JSON.parse(localStorage.getItem('currentUser')).progress.ex3Score;
  }
}
```

Лістинг файлу «ex1.component.html»:

```
<section id="ex1" class="exercise">
  <div class="exercise__content">
    <div class="column column-left">
      <app-task1 [detail]="currentExercise"></app-task1>
    </div>
    <div class="column column-right">
      <app-answer1 [detail]="currentExercise"
(answerChecked)="onAnswerChecked()"></app-answer1>
    </div>
  </div>
  <div class="sidebar-wrapper">
    <app-sidebar [taskList]="taskList" (taskChange)="onTaskChange($event)"></app-
sidebar>
  </div>
</section>
```

Лістинг файлу «ex1.component.ts»:

```
import { Component, OnInit } from '@angular/core';
import { ExerciseModel, SelectedTaskModel, TaskLevelModel, CodeEditorModel } from
'@/models';
import { TaskService } from '@/_services/task.service';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-ex1',
  templateUrl: './ex1.component.html',
  styleUrls: ['./ex1.component.scss']
})
export class Ex1Component implements OnInit {
  show: Boolean = false;
  taskList: TaskLevelModel[];
  currentExercise: ExerciseModel;
  tests: any;

  constructor(private taskService: TaskService, private toastr: ToastrService) { }

  ngOnInit() {
```

```
    let currentUser = JSON.parse(localStorage.getItem('currentUser'));
    this.taskService.getEx1TaskList().subscribe(data => {
      this.taskList = data;
      // condition on last available task
      let lastTasks = this.taskList[this.taskList.length - 1].tasks;
      if (currentUser.progress.ex1Score == lastTasks[lastTasks.length - 1].id) {
        this.currentExercise = lastTasks[lastTasks.length - 1];
      } else {
        this.currentExercise =
this.taskList[currentUser.progress.ex1Level].tasks[currentUser.progress.ex1Score];
      }
    });
  }

  onTaskChange(model: SelectedTaskModel) {
    const level = this.taskList.find(x => x.id === model.levelId);
    this.currentExercise = level.tasks.find(t => t.id === model.taskId);
    this.tests = this.currentExercise.tests;
  }

  onAnswerChecked() {
    const taskId = this.currentExercise.id;
    const nextTaskId = taskId + 1;
    const lastTasksLength = this.taskList[this.taskList.length - 1].tasks.length;
    const lastTaskId = this.taskList[this.taskList.length - 1].tasks[lastTasksLength -1].id;
    // it was last task available
    if (nextTaskId > lastTaskId) {
      this.toastr.info("Ви виконали останнє завдання");
      let currentUser = JSON.parse(localStorage.getItem('currentUser'));
      currentUser.progress.ex1Score = taskId;
      localStorage.setItem('currentUser', JSON.stringify(currentUser));
      return;
    }
    // redirect to next task
    let taskLevel;

    for (let i = 0; i < this.taskList.length; i++) {
      let levelLastTaskId = this.taskList[i].tasks[this.taskList[i].tasks.length - 1].id;
      if (nextTaskId <= levelLastTaskId) {
        taskLevel = i;
        this.currentExercise = this.taskList[taskLevel].tasks.find(t => t.id === nextTaskId);
        break;
      }
    }

    // memorize in localStorage
```

```
    let currentUser = JSON.parse(localStorage.getItem('currentUser'));
    currentUser.progress.ex1Score = taskId;
    currentUser.progress.ex1Level = taskLevel;
    localStorage.setItem('currentUser', JSON.stringify(currentUser));
  }
}
```

Лістинг файлу «task1.component.html»:

```
<div class="description">
  <h1 class="description__title">
    {{detail?.name}}
  </h1>
  <div [innerHtml] = detail?.description></div>
  <hr>
  <!-- Check list -->
  <div class="check-list">
    <h4>Потрібно виконати:</h4>
    <ul>
      <li *ngFor="let challenge of detail?.challenges">{{challenge.title}}</li>
    </ul>
  </div>
</div>
```

Лістинг файлу «task1.component.ts»:

```
import { Component, Input, OnInit } from '@angular/core';
import { ExerciseModel } from '@/models';

@Component({
  selector: 'app-task1',
  templateUrl: './task1.component.html',
  styleUrls: ['./task1.component.scss']
})

export class Task1Component implements OnInit {

  @Input() detail: any;

  constructor() { }

  ngOnInit() {
  }
}
```

Лістинг файлу «answer1.component.html»:

```
<div class="editor-wrapper">
```

```
  <ace-editor #editor (textChanged)="onChange($event)"
            [autoUpdateContent]="true"
            [mode]="'javascript'"
            [options]="options"
            [readOnly]="false"
            [theme]="'iplastic'"
            [(text)]="detail?.codeEditor[0].text" style="min-height: 93.5vh; width:100%;
overflow: auto;">
  </ace-editor>
</div>

<!-- check answer -->
<button (click)="ex1Verification()" class="check-answer" title="Виконати перевірку">
  <img src="assets/img/play-icon.png" alt="Перевірити">
</button>
```

Лістинг файлу «answer1.component.ts»:

```
import { Component, Input, Output, EventEmitter, OnInit, ViewChild, AfterViewInit }
from '@angular/core';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-answer1',
  templateUrl: './answer1.component.html',
  styleUrls: ['./answer1.component.scss']
})

export class Answer1Component implements OnInit, AfterViewInit {
  editorCode: string;

  @Input() detail: any;

  @Output() answerChecked = new EventEmitter();

  options:any = { maxLines: 1000, fontSize: '22px', printMargin: false };

  @ViewChild('editor') editor;

  constructor(private toastr: ToastrService) { }

  ngOnInit() {
  }

  ngAfterViewInit() {
    this.editor.getEditor().setOptions({
```

```
    enableBasicAutocompletion: true
   });
   this.editor.getEditor().commands.addCommand({
    name: 'showOtherCompletions',
    bindKey: 'Ctrl-.',
    exec: function (editor) {
    }
   });
  }

  onChange(code) {
   this.editorCode = code;
  }

  // verification of the answer

  ex1Verification(){
   const tests = this.detail.tests;

   for (let i = 0; i < tests.length; i++) {
    if (!this.editorCode) {
     this.toastr.error("Ви допустили помилку. Спробуйте ще раз");
     return;
    }
    if (this.editorCode.indexOf(tests[i].value) == -1) {
     this.toastr.error("Ви допустили помилку. Спробуйте ще раз");
     return;
    }
   }

   this.toastr.success("Відповідь правильна. Переходимо до наступного завдання");
   this.answerChecked.emit("Answer correct");
  }
}
```

Лістинг файлу «task2.component.html»:

```
<div class="description">
  <h1 (click)="showDetail()" class="description__title">
   {{detail?.name}}
  </h1>
  <!-- question [innerHMTL]-->
  <div [innerHTML] = "detail?.question" class="question-block"></div>

  <div class="editor-wrapper">
   <ace-editor #editor
```

```
    [mode]="'javascript'"
    [options]="options"
    [readOnly]="true"
    [theme]="'iplastic'"
    [(text)]="detail?.codeEditor[0].text"
    style="min-height: 72vh; width:100%; overflow: auto;"
   ></ace-editor>
  </div>
</div>
```

Лістинг файлу «task2.component.ts»:

```
import { Component, OnInit, Input, ViewChild, AfterViewInit } from '@angular/core';

@Component({
  selector: 'app-task2',
  templateUrl: './task2.component.html',
  styleUrls: ['./task2.component.scss']
})
export class Task2Component implements OnInit {

  options: any = {
    maxLines: 1000,
    fontSize: '22px',
    printMargin: false
  };

  @Input() detail: any;

  @ViewChild('editor') editor: any;

  constructor() { }

  ngOnInit() {
  }

  showDetail() {
    console.log(this.detail);
  }
}
```

Лістинг файлу «answer2.component.html»:

```
<div class="answer">
  <h4>Введіть вашу відповідь:</h4>
  <input [(ngModel)]="userAnswer" type="text">
  <!-- check answer -->
```

```
  <button (click)="ex2Verification()" class="check-answer" title="Виконати перевірку">
    <img src="assets/img/play-icon.png" alt="Перевірити">
  </button>
</div>
```

Лістинг файлу «answer2.component.ts»:

```
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-answer2',
  templateUrl: './answer2.component.html',
  styleUrls: ['./answer2.component.scss']
})
export class Answer2Component implements OnInit {
  userAnswer: string = "";

  @Input() detail: any;

  @Output() answerChecked = new EventEmitter();

  constructor(private toastr: ToastrService) { }

  ngOnInit() {
  }

  ex2Verification() {
    if (this.userAnswer == this.detail.answer) {
      this.answerChecked.emit("Answer correct");
      this.toastr.success("Відповідь правильна. Переходимо до наступного завдання");
      this.userAnswer = "";
    } else {
      this.toastr.error("Ви допустили помилку. Спробуйте ще раз");
    }
    // console.log(this.answer);
  }
}
```

Лістинг файлу «task3.component.html»:

```
<div class="description">
  <h1 class="description__title">{{detail?.name}}
  </h1>
  <div [innerHTML]="detail?.question" class="description__content"></div>
</div>
```

Лістинг файлу «task3.component.ts»:

```
import { Component, Input, OnInit } from '@angular/core';

@Component({
  selector: 'app-task3',
  templateUrl: './task3.component.html',
  styleUrls: ['./task3.component.scss']
})
export class Task3Component implements OnInit {

  @Input() detail: any;

  constructor() { }

  ngOnInit() {
  }
}
```

Лістинг файлу «answer3.component.html»:

```
<div class="answer">
  <h4>Виберіть правильну відповідь:</h4>

  <ul class="choises">
    <li *ngFor="let item of detail?.choises" (click)="onSelect(item)" [class.active] = "item.id
== activeItem">{{item.name}}</li>
  </ul>

  <!-- check answer -->
  <button (click)="ex3Verification()" class="check-answer" title="Виконати перевірку">
    <img src="assets/img/play-icon.png" alt="Перевірити">
  </button>
</div>
```

Лістинг файлу «answer3.component.ts»:

```
import { Component, Input, Output, EventEmitter, OnInit } from '@angular/core';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-answer3',
  templateUrl: './answer3.component.html',
  styleUrls: ['./answer3.component.scss']
})
export class Answer3Component implements OnInit {

  activeItem: Number = 0;
```

```
  @Input() detail: any;

  @Output() answerChecked = new EventEmitter();

  constructor(private toastr: ToastrService) { }

  ngOnInit() {
  }

  onSelect(item) {
    this.activeItem = item.id;
  }

  ex3Verification() {
    if (this.activeItem == this.detail.answer) {
      this.toastr.success("Відповідь правильна. Переходимо до наступного завдання");
      this.answerChecked.emit("Answer correct");
      this.activeItem = 0;
    } else {
      this.toastr.error("Ви допустили помилку. Спробуйте ще раз");
    }
  }
}
```

Лістинг файлу «cabinet.component.html»:

```
<section class="cabinet">
  <div class="container">
    <h1>Налаштування</h1>
    <div class="setting-group">
      <div class="setting">
        <button (click)="resetAllProgress()">Обнулити весь прогрес</button>
      </div>
      <div class="setting">
        <button (click)="resetEx1Progress()">Обнулити вправу "Тренуй себе"</button>
      </div>
      <div class="setting">
        <button (click)="resetEx2Progress()">Обнулити вправу "Визнач
результат"</button>
      </div>
      <div class="setting">
        <button (click)="resetEx3Progress()">Обнулити вправу "Перевір
знання"</button>
      </div>
    </div>
```

```
  </div>
</section>
<app-footer></app-footer>
```

Лістинг файлу «cabinet.component.ts»:

```
import { Component, OnInit } from '@angular/core';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-cabinet',
  templateUrl: './cabinet.component.html',
  styleUrls: ['./cabinet.component.scss']
})
export class CabinetComponent implements OnInit {


  constructor(private toastr: ToastrService) { }

  ngOnInit() {
  }

  resetAllProgress() {
    let currentUser = JSON.parse(localStorage.getItem('currentUser'));
    currentUser.progress = {
      ex1Level: 0,
      ex1Score: 0,
      ex2Level: 0,
      ex2Score: 0,
      ex3Level: 0,
      ex3Score: 0
    }
    localStorage.setItem('currentUser', JSON.stringify(currentUser));
    this.toastr.success("Прогрес всіх вправ успішно оновлено");
  }

  resetEx1Progress() {
    let currentUser = JSON.parse(localStorage.getItem('currentUser'));
    currentUser.progress.ex1Level = 0;
    currentUser.progress.ex1Score = 0;
    localStorage.setItem('currentUser', JSON.stringify(currentUser));
    this.toastr.success("Прогрес вправи \"Тренуй себе\" успішно оновлено");
  }

  resetEx2Progress() {
    let currentUser = JSON.parse(localStorage.getItem('currentUser'));
```

```
    currentUser.progress.ex2Level = 0;
    currentUser.progress.ex2Score = 0;
    localStorage.setItem('currentUser', JSON.stringify(currentUser));
    this.toastr.success("Прогрес вправи \"Визнач результат\" успішно оновлено");
  }

  resetEx3Progress() {
    let currentUser = JSON.parse(localStorage.getItem('currentUser'));
    currentUser.progress.ex3Level = 0;
    currentUser.progress.ex3Score = 0;
    localStorage.setItem('currentUser', JSON.stringify(currentUser));
    this.toastr.success("Прогрес вправи \"Перевір знання\" успішно оновлено");
  }
}
```

Лістинг файлу «additionally.component.html»:

```
<section id="additionally" class="additionally">
  <div class="container">
    <h1>Додаткові ресурси для вивчення JavaScript</h1>
    <div class="content">
      <p>
        <a href="https://learn.javascript.ru">JavaScript довідник</a> – веб-ресурс, що описує функції та конструкції мови JavaScript на російській мові. На ресурсі є багато статей про JavaScript та його примінення.
      </p>
      <p>
        <a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript">A re-introduction to JavaScript (JS tutorial)</a> – чудова стаття англійською мовою від команди розробників Mozilla Foundation.
      </p>
      <p>
        <a href="https://learn.freecodecamp.org/">FreeCodeCamp</a> – безкоштовний веб-ресурс для вивчення мови JavaScript з великою кількістю завдань починаючи від основ і закінчуючи фреймворками та Node.js. Чудове місце попрактикуватися в написанні коду.
      </p>
      <p>
        <a href="https://www.codewars.com">Codewars</a> – веб-ресурс, що підходить людям, які бажають позмагатися в написанні коду та покращити свої вміння.
      </p>
    </div>
  </div>
</section>
<app-footer></app-footer>
```

Лістинг файлу «styles.scss»:

```
* {
  box-sizing: border-box;
}

body {
  margin: 0;
  font-family: 'Arial', sans-serif;
  line-height: 1;
  height: 100%;
}

h1, h2, h3, h4, h5, h6 {
  margin: 0;
}

ul {
  padding: 0;
  margin: 0;
  list-style: none;
}

p {
  margin-top: 0;
  margin-bottom: 0.5em;
}

a, a:hover {
  color: #1010ea;
  cursor: pointer;
  text-decoration: none;
}

.container {
  padding: 0 15px;
  margin: 0 auto;
  width: 100%;
  max-width: 1200px;
}

.hidden {
  display: none !important;
}

.form-group {
```

```
  margin-bottom: 25px;
  font-size: 14px;
  display: flex;
  flex-direction: column;
  label {
    margin-bottom: 10px;
    color: #1A1A1A;
    span {
      color: red;
    }
  }
  input {
    padding: 5px 10px;
    height: 35px;
    width: 100%;
    max-width: 318px;
    border: 1px solid #CFDBCF;
    border-radius: 3px;
    background: #fff;
  }
}

.form-buttons {
  font-size: 14px;
  button {
    padding: 10px 21px;
    margin-right: 15px;
    color: #fff;
    font-weight: 700;
    background: #007BFF;
    border: none;
    border-radius: 3px;
    cursor: pointer;
    transition: all .3s ease;
    &:hover {
      background: #0059FF;
    }
  }
  a {
    color: #007BFF;
    &:hover {
      color: #007BFF;
      text-decoration: underline;
    }
  }
}
```

```
/* exercises */
$ex1_padding: 20px;

.exercise {
  padding-right: 60px;
  background-color: #dbdcf2;
  height: 100vh;
  display: flex;
  justify-content: space-between;
  position: relative;
  .exercise__content {
    padding: $ex1_padding;
    width: 100%;
    display: flex;
    justify-content: space-between;
  }
  .column {
    width: 50%;
  }
  .column-left {
    padding-right: $ex1_padding - 10px;
  }
  .column-right {
    padding-left: $ex1_padding - 10px;
  }
}

// sidebar
.sidebar-wrapper {
  width: 60px;
  height: 100%;
  position: absolute;
  top: 0;
  right: 0;
}

// button check answer
.check-answer {
  display: block;
  padding: 15px 14px;
  width: 60px;
  height: 58px;
  border: none;
  background: transparent;
  outline: none;
```

```
  cursor: pointer;
  transition: all .3s ease;
  position: fixed;
  right: 0;
  bottom: 0;
  z-index: 100;
  &:hover {
    background: #d68c15;
  }
  img {
    display: block;
    width: 100%;
    height: auto;
  }
}

// description
.description {
  padding: 20px;
  height: 100%;
  font-size: 16px;
  background: #eee;
  overflow: auto;
  .description__title {
    margin-bottom: 20px;
  }
  p {
    margin-bottom: 15px;
    font-size: inherit;
    line-height: 1.3em;
    i {
      color: #750000;
      font-style: normal;
    }
  }
  blockquote {
    padding: 10px 20px;
    margin: 0 0 15px;
    color: #750000;
    line-height: 1.4em;
    font-size: inherit;
    background-color: #e1e1e1;
  }
  .check-list {
    padding: 15px 0 30px;
    h4 {
```

```
      margin-bottom: 10px;
    }
    li {
      padding: 10px;
      margin-bottom: 10px;
      background: #e1e1e1;
      border-radius: 10px;
      &:last-child {
        margin-bottom: 0;
      }
    }
  }
}

.answer {
  height: 100%;
  background: #eee;
  overflow: auto;
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: center;
  h4 {
    margin: 20px 0;
    font-size: 20px;
  }
  input[type="text"] {
    display: block;
    width: 100%;
    max-width: 300px;
    font-size: 20px;
  }
}
```

Лістинги файлів back-end частини

Лістинг файлу «package.json»:

```
{
  "main": "main.js",
  "nodemonConfig": {
    "watch": [
      "app"
    ],
    "ext": "js",
    "ignore": [
      "js-tower.json"
    ],
    "delay": "3"
  },
  "scripts": {
    "dev": "nodemon main.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "dependencies": {
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "express": "^4.16.4",
    "jsonwebtoken": "^8.5.1",
    "lokijs": "^1.5.6"
  },
  "devDependencies": {
    "nodemon": "^1.19.0"
  }
}
```

Лістинг файлу «main.js»:

```
const app = require('./app/server');

app.listen(8080, function(){
  console.log('you may use rest api at port 8080');
});
```

Лістинг файлу «server.js»:

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors')
const app = express();

const corsOptions = {
   origin: 'http://localhost:4200',
```

```
  optionsSuccessStatus: 200 // some legacy browsers (IE11, various SmartTVs) choke on
204
};

app.use(cors(corsOptions));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));

require('./database');
require('./routes')(app);

module.exports = app;
```

Лістинг файлу «routes.js»:

```
const authApi = require('./controllers/authApi');
const taskApi = require('./controllers/taskApi');

module.exports = function(app){
 app.get('/api/user/list', authApi.userList);
 app.post('/api/user/score', authApi.saveScore);

 app.post('/api/auth/login', authApi.login);
 app.post('/api/auth/logout', authApi.logout);
 app.post('/api/auth/register', authApi.register);

 app.get('/api/ex1/task/list', taskApi.ex1TaskList);
 app.get('/api/ex2/task/list', taskApi.ex2TaskList);
 app.get('/api/ex3/task/list', taskApi.ex3TaskList);
};
```

Лістинг файлу «authApi.js»:

```
const db = require('../util/db');
const jwtoken = require('jsonwebtoken');
const jwtSecret = 'FACE7CDC-7643-4AC1-861D-42ED99CCCC32';

exports.userList = function(req, res){
  db.loadDatabase({}, function () {
    res.status(200).json({data: db.getCollection('users').data});
  });
};

exports.login = function(req, res) {
  if(!req.body) return res.sendStatus(400);
  const query = {login: req.body.username, password: req.body.password};
  db.loadDatabase({}, function () {
```

```
        const data = db.getCollection('users');
        const result = data.findOne({'login': query.login, 'password': query.password});
        if (result === null) {
            res.status(500).send({error: 'An error has occured'});
        } else {
            const token = jwtoken.sign(result.login, jwtSecret);
          res.status(200).json({ data: { id: result.id, login: result.login, token: token, progress:
result.progress }});
        }
    });
};

exports.logout = function(req, res){
    if(!req.body) return res.sendStatus(400);
    const query = {token: req.body.token, score: req.body.score};
    db.loadDatabase({}, function () {
        const data = db.getCollection('users');
        const result = data.findOne({'token': query.token});
        if (result === null) {
            res.status(500).send({error: 'An error has occured'});
        } else {
            result.update((user) => {user.score = query.score});
            res.status(200).json({data: result});
        }
    });
};

exports.register = function(req, res){
    if(!req.body) return res.sendStatus(400);
    const query = { login: req.body.username, password: req.body.password, token:
req.body.password + '-jwt-token', score: 0 };
    db.loadDatabase({}, function () {
        const data = db.getCollection('users');
        data.insert(query);
        db.saveDatabase();
        const result = data.findOne({'login': query.login, 'password': query.password});
        if (result === null) {
            res.status(500).send({error: 'An error has occured'});
        } else {
            res.status(200).json({data: result});
        }
    });
};

exports.saveScore = function(req, res){
    if(!req.body) return res.sendStatus(400);
```

```
    const query = {token: req.body.token, score: req.body.score};
    db.loadDatabase({}, function () {
        const data = db.getCollection('users');
        const result = data.findOne({'token': query.token});
        if (result === null) {
            res.status(500).send({error: 'An error has occured'});
        } else {
            result.update((user) => {user.score = query.score});
            res.status(200).json({data: result});
        }
    });
};
```

Лістинг файлу «taskApi.js»:

```
const db = require('../util/db');

exports.ex1TaskList = function(req, res){
  db.loadDatabase({}, function () {
      res.status(200).json({data: db.getCollection('ex1').data});
  });
};

exports.ex2TaskList = function (req, res) {
  db.loadDatabase({}, function () {
    res.status(200).json({ data: db.getCollection('ex2').data });
  });
};

exports.ex3TaskList = function (req, res) {
  db.loadDatabase({}, function () {
    res.status(200).json({ data: db.getCollection('ex3').data });
  });
};

exports.exercise = function(req, res){
    let query = req.query.taskId;
    if(query) {
        let id = parseInt(query);
        db.loadDatabase({}, function () {
            res.status(200).json({data: db.getCollection('exercises').find({taskId: id})});
        });
    } else {
        res.status(500).send({error: 'An error has occured'});
    }
};
```

Лістинг файлу «database.js»:

```
const loki = require('lokijs');

const db = new loki('js-tower.json');

db.addCollection('users').insert([
  { id: 1, login: 'test', password: 'test', token: 'test-jwt-token',
    progress: {
      ex1Level: 0,
      ex1Score: 0,
      ex2Level: 0,
      ex2Score: 0,
      ex3Level: 0,
      ex3Score: 0
    }
  },
  { id: 2, login: 'demo', password: 'demo', token: 'demo-jwt-token',
    progress: {
      ex1Level: 0,
      ex1Score: 0,
      ex2Level: 0,
      ex2Score: 0,
      ex3Level: 0,
      ex3Score: 0
    }
  }
]);

/* Collection for ex1 */
db.addCollection('ex1').insert([
  // level 1 - EX1
  {
    id: 1,
    name: "Рівень 1",
    tasks: [
      // task 1 - EX1
      {
        id: 1,
        name: "Коментарі",
        description: `<p>Коментарі - це спеціальний синтаксис в JavaScript, який
дозволяє ігнорувати один або більше рядків коду. Частіше всього коментарі
використовуються для того, щоб залишити текстові помітки в коді, які допоможуть
швидко розібратися в тому, що цей код робить.</p>
        <p>Написати коментар можна двома способами:</p>
```

```
      <p>За допомогою <i>//</i>, буде проігнорований весь подальший текст в цьому рядку.</p>
      <blockquote>// Це простий коментар</blockquote>
      <p>За допомогою <i>/*</i> і <i>*/</i>, буде проігнорований весь внутрішній текст.</p>
      <blockquote>/* Це коментар<br>на декілька<br>рядків*/</blockquote>
      <p>Коли ти пишеш код, намагайся часто коментувати його, це допоможе швидко в ньому розібратися</p>`,
      codeEditor: [{id: 1, text: ``}],
      challenges: [
        { id: 1, title: "Створи простий коментар" },
        { id: 2, title: "Створи багаторядковий коментар" }
      ],
      tests: [
        { id: 1, value: "//" },
        { id: 2, value: "/*" },
        { id: 3, value: "*/" }
      ]
    },
    // end task 1
    // task 2 - EX1
    {
      id: 2,
      name: "Об'явлення змінних",
      description: `<p>В JavaScript існують сім типів даних: <i>undefined</i>, <i>null</i>, <i>boolean</i>, <i>string</i>, <i>symbol</i>, <i>number</i> та <i>object</i>.</p>
      <p>Самий простий спосіб об'явити змінну - за допомогою ключового слова <i>var</i>. Спочатку ми вказуємо ключове слово <i>var</i>, далі назву змінної та крапку з комою в кінці.</p>
      <blockquote>var myVariable;</blockquote>
      <p>Спробуй створити змінну з назвою <i>myName</i>.</p>`,
      codeEditor: [{ id: 1, text: ``}],
      challenges: [
        { id: 1, title: "Об'яви змінну з назвою myName" },
      ],
      tests: [
        { id: 1, value: "var myName;" }
      ]
    },
    // end task 2
    // task 3 - EX1
    {
      id: 3,
      name: "Ключові слова let і const",
```

```
      description: `<p>В стандарті ETMAScript 6 (ES6) запровадженому в 2015 році були додані нові можливості мови.</p>
      <p>Згідно до нового стандарту краще об'являти змінну за допомогою ключових слів <i>let</i> або <i>const</i>.</p>
      <p>Ключове слово let працює схожим чином як і var, але областю визначення let змінної є не лише функції але й будь які логічні блоки виділені фігурними дужками</p>
      <p>Ключове слово const відрізняється від let тим, що після присвоєння змінній якогось значення, це значення більше не можна змінювати, воно стає константою</p>
      <p>Приклад:</p>
      <blockquote>
        let variableOne = 20;<br>
        const variableTwo = 30;
      </blockquote>`,
      codeEditor: [{ id: 1, text: `` }],
      challenges: [
        { id: 1, title: "Об'яви змінну за допомогою ключового слова let" },
        { id: 1, title: "Об'яви змінну за допомогою ключового слова const" }
      ],
      tests: [
        { id: 1, value: "let" },
        { id: 2, value: "const" }
      ]
    },
    // end task 3
    // task 4 - EX1
    {
      id: 4,
      name: "Завдання",
      description: ``,
      codeEditor: [{ id: 1, text: `` }],
      challenges: [
        { id: 1, title: "" }
      ],
      tests: [
        { id: 1, value: "" }
      ]
    }
    // end task 4
  ]
 },
 // end level 1 - EX1
 // level 2 - EX1
 {
  id: 2,
```

```
name: "Рівень 2",
tasks: [
 // task 5 - EX1
 {
  id: 5,
  name: "Завдання",
  description: ``,
  codeEditor: [{ id: 1, text: `` }],
  challenges: [
   { id: 1, title: "" }
  ],
  tests: [
   { id: 1, value: "" }
  ]
 },
 // end task 5
 // task 6 - EX1
 {
  id: 6,
  name: "Завдання",
  description: ``,
  codeEditor: [{ id: 1, text: `` }],
  challenges: [
   { id: 1, title: "" }
  ],
  tests: [
   { id: 1, value: "" }
  ]
 },
 // end task 6
 // task 7 - EX1
 {
  id: 7,
  name: "Завдання",
  description: ``,
  codeEditor: [{ id: 1, text: `` }],
  challenges: [
   { id: 1, title: "" }
  ],
  tests: [
   { id: 1, value: "" }
  ]
 },
 // end task 7
 // task 8 - EX1
 {
```

```
      id: 8,
      name: "Завдання",
      description: ``,
      codeEditor: [{ id: 1, text: `` }],
      challenges: [
       { id: 1, title: "" }
      ],
      tests: [
       { id: 1, value: "" }
      ]
     },
     // end task 8
     // task 9 - EX1
     {
      id: 9,
      name: "Завдання",
      description: ``,
      codeEditor: [{ id: 1, text: `` }],
      challenges: [
       { id: 1, title: "" }
      ],
      tests: [
       { id: 1, value: "" }
      ]
     },
     // end task 9
     // task 10 - EX1
     {
      id: 10,
      name: "Завдання",
      description: ``,
      codeEditor: [{ id: 1, text: `` }],
      challenges: [
       { id: 1, title: "" }
      ],
      tests: [
       { id: 1, value: "" }
      ]
     }
     // end task 10
    ]
  }
  // end level 2 - EX1
]);

/* Collection for ex2 */
```

```
db.addCollection('ex2').insert([
  // level 1 - EX2
  {
    id: 1,
    name: "Рівень 1",
    tasks: [
      // task 1 - EX2
      {
        id: 1,
        name: "Завдання 1",
        question: `<p>Яке значення буде виведено в консоль?</p>`,
        codeEditor: [
          {
            id: 1,
            text: `function sayHello(name) {
    return name.toUpperCase();
}

let yourSecondName = "Colt";
sayHello(yourSecondName);
console.log(sayHello("John") + " " + sayHello(yourSecondName));`
          }
        ],
        answer: "JOHN GOLT"
      },
      // end task 1
      // task 2 - EX2
      {
        id: 2,
        name: "Завдання 2",
        question: `<p>Яке значення виведе в консоль console.log, що спрацює
останнім?</p>`,
        codeEditor: [
          {
            id: 1,
            text: `function noReturn() {
  var sum = 0;
  for(var i = 0; i < 10; i++) {
    sum += i;
  }
  console.log(sum);
}
console.log(noReturn());`
          }
        ],
        answer: "undefined"
```

```
    },
    // end task 2
    // task 3 - EX2
    {
      id: 3,
      name: "Завдання 3",
      question: `<p>Яке значення виведе в консоль?</p>`,
      codeEditor: [
        {
          id: 1,
          text: `const str = "101-1";
console.log(parseInt(str, 2));`
        }
      ],
      answer: "5"
    },
    // end task 3
    // task 4 - EX2
    {
      id: 4,
      name: "Завдання 4",
      question: `<p>Яке значення виведе в консоль?</p>`,
      codeEditor: [
        {
          id: 1,
          text: `console.log(parseInt('hello', 10));`
        }
      ],
      answer: "NaN"
    },
    // end task 4
    // task 5 - EX2
    {
      id: 5,
      name: "Завдання 5",
      question: `<p>Яке значення виведе в консоль?</p>`,
      codeEditor: [
        {
          id: 1,
          text: `let one = 1;
let zero = 0;
console.log(one/zero + one);`
        }
      ],
      answer: "Infinity"
    }
```

```
    // end task 5
   ]
  },
  // end level 1 - EX2
  // level 2 - EX2
  {
   id: 2,
   name: "Рівень 2",
   tasks: [
    // task 6 - EX2
    {
     id: 6,
     name: "Завдання 6",
     question: `<p>Яке значення отримає змінна name?</p>`,
     codeEditor: [
      {
       id: 1,
       text: `let name = 'kittens';
if (name == 'puppies') {
  name += ' woof';
} else if (name == 'kittens') {
  name += ' meow';
} else {
  name += '!';
}`
      }
     ],
     answer: "kittens meow"
    },
    // end task 6
    // task 7 - EX2
    {
     id: 7,
     name: "Завдання 7",
     question: `<p>Яке значення виведе в консоль?</p>`,
     codeEditor: [
      {
       id: 1,
       text: ``
      }
     ],
     answer: ""
    },
    // end task 7
   ]
  }
```

```
  // end level 2 - EX2
]);

/* Collection for ex3 */
db.addCollection('ex3').insert([
  // level 1 - EX3
  {
    id: 1,
    name: "Рівень 1",
    tasks: [
      // task 1 - EX3
      {
        id: 1,
        name: "Завдання 1",
        question: `<p>Скільки типів даних в JavaScript?</p>`,
        choises: [
          { id: 1, name: "П'ять" },
          { id: 2, name: "Шість" },
          { id: 3, name: "Сім" },
          { id: 4, name: "Вісім" }
        ],
        answer: 3
      },
      // end task 1
      // task 2 - EX3
      {
        id: 2,
        name: "Завдання 2",
        question: `<p>Якого типу NaN?</p>`,
        choises: [
          { id: 1, name: "Object" },
          { id: 2, name: "Number" },
          { id: 3, name: "String" },
          { id: 4, name: "null" },
          { id: 5, name: "undefined" }
        ],
        answer: 2
      },
      // end task 2
      // task 3 - EX3
      {
        id: 3,
        name: "Завдання 3",
        question: `<p></p>`,
        choises: [
          { id: 1, name: "Object" },
```

```
      { id: 2, name: "Number" },
      { id: 3, name: "String" },
      { id: 4, name: "null" },
      { id: 5, name: "undefined" }
    ],
    answer: 2353532
  }
  // end task 3
 ]
},
// end level 1
// level 2 - EX3
{
 id: 2,
 name: "Рівень 2",
 tasks: [
  // task 4 - EX3
  {
   id: 4,
   name: "Завдання 4",
   question: `<p></p>`,
   choises: [
    { id: 1, name: "Object" },
    { id: 2, name: "Number" },
    { id: 3, name: "String" },
    { id: 4, name: "null" },
    { id: 5, name: "undefined" }
   ],
   answer: 2353532
  },
  // end task 4
  // task 5 - EX3
  {
   id: 5,
   name: "Завдання 5",
   question: `<p></p>`,
   choises: [
    { id: 1, name: "Object" },
    { id: 2, name: "Number" },
    { id: 3, name: "String" },
    { id: 4, name: "null" },
    { id: 5, name: "undefined" }
   ],
   answer: 2353532
  },
  // end task 5
```

```
    // task 6 - EX3
    {
      id: 6,
      name: "Завдання 6",
      question: `<p></p>`,
      choises: [
        { id: 1, name: "Object" },
        { id: 2, name: "Number" },
        { id: 3, name: "String" },
        { id: 4, name: "null" },
        { id: 5, name: "undefined" }
      ],
      answer: 2353532
    }
    // end task 6
  ]
 },
 // end level 2
]);

db.saveDatabase();
```

Лістинг файлу «main.js»: